Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Cabina de Realidad Virtual en Unity para operar un sistema multi-UAV en ROS en tareas de inspección de turbinas de viento

Autor: Emilio Jesús Marín de la Rosa

Tutor: Jesús Capitán Fernández, Damián Jesús Pérez Morales

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Cabina de Realidad Virtual en Unity para operar un sistema multi-UAV en ROS en tareas de inspección de turbinas de viento

Autor:
Emilio Jesús Marín de la Rosa

Tutor:
Jesús Capitán Fernández, Damián Jesús Pérez Morales
Profesor Titular, Contratado Investigador

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Grado: Cabina de Realidad Virtual en Unity para operar un sistema multi-UAV en ROS en tareas de inspección de turbinas de viento

Autor: Emilio Jesús Marín de la Rosa
Tutor: Jesús Capitán Fernández, Damián Jesús Pérez Morales

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

A mi familia.

A mi pareja.

Agradecimientos

Con este Trabajo Fin de Grado se pone fin a una de las etapas más importantes de mi vida. Quisiera agradecer con estas palabras a todas las personas que me han acompañado y ayudado a llegar hasta aquí.

Para comenzar, me gustaría agradecer a mi tutor Jesús Capitán Fernández por ayudarme y guiarme en este proyecto tan novedoso e interesante. A Damián Jesús Pérez Morales por su inestimable ayuda y amabilidad a lo largo de todo el proyecto. Sin él, este trabajo no hubiera sido posible.

En especial, a mis padres Emilio y Rosalía, por haberme ayudado siempre y confiar plenamente en mí. Por su amor y apoyo incondicional. Nada de esto hubiera sido posible sin ellos.

A Carmen, por formar parte de mi vida y por estar en los buenos, pero sobre todo, en los malos momentos. Por ser un pilar fundamental en mi vida y hacerme feliz. Por tu cariño siempre. Gracias por estar aquella tarde en aquel banco conmigo.

Por último, a mis amigos y compañeros del Grado. Por su ayuda durante este tiempo y porque sin ellos esta experiencia no hubiera sido igual.

Gracias por todo.

Emilio Jesús Marín de la Rosa

Sevilla, 2022

Resumen

En este proyecto se presenta el desarrollo de una Cabina de Realidad Virtual con el motor gráfico Unity. Esta Cabina se usa para operar sistemas multi-UAV en tareas de inspección de turbinas de viento y se ejecuta en unas gafas de Realidad Virtual.

Esta tarea de inspección se ejecuta en ROS, un framework de desarrollo de robots. Esto requerirá desarrollar comunicaciones entre Unity y ROS. Para ello se usará ROS#, que es un conjunto de librerías, software y herramientas para comunicar ROS con Unity.

La Cabina Virtual cuenta con unas pantallas que presentan las vistas de las cámaras de los UAV. Además, posee una mesa de status y control donde se puede visualizar información valiosa de la tarea de inspección.

También, se crea un mapa en 3D que emula el entorno real dentro del entorno virtual. Esto permite al usuario tener una mejor perspectiva de la misión.

Po último, se desarrollan los controles necesarios para poder hacer uso de la Cabina. Estos controles se implementan en unos mandos de Realidad Virtual.

Abstract

This project presents the development of a Virtual Reality Cockpit with the Unity graphics engine. This Cockpit is used to operate multi-UAV systems in wind turbine inspection tasks and is executed in Virtual Reality glasses.

This inspection task runs on ROS, a robot development framework. This will require developing communications between Unity and ROS. ROS# will be used for this, which is a set of libraries, software and tools to communicate ROS with Unity.

The Virtual Cockpit has screens that present the camera views of the UAVs. In addition, it has a status and control table where valuable information of the inspection task is displayed.

Also, a 3D map is created. This map emulates the real environment within the virtual environment. This allows the user to have a better perspective of the mission.

Finally, the necessary controls are developed to be able to use the Cockpit. These controls are implemented in Virtual Reality controllers.

Índice abreviado

Índice

1 Introducción

Se expondrá la motivación de este proyecto y los objetivos a cumplir con el mismo.

1.1 Motivación

Durante muchos años, la Realidad Virtual ha sido una tecnología desconocida para gran parte de la sociedad. En parte, esto es debido a que la Realidad Virtual es una tecnología aún en auge. A pesar de ello, es una de las tecnologías más avanzadas de la actualidad y promete ser una tecnología de futuro y, que sin duda, desempeñará un papel fundamental en la sociedad.

Las aplicaciones de la Realidad Virtual son muy variadas como se verá en la Sección 2.3. Se usa en aplicaciones de entretenimiento, salud, seguridad, educación, etc. Posee propiedades que otras tecnologías no tienen y que la hacen ideal para muchas aplicaciones. Además, al no estar completamente desarrollada, no se puede ser consciente de todas las oportunidades que aún están por descubrir.

Como se abordará en la Sección 2.1, las energías renovables se posicionan cada vez más como la alternativa ganadora en el ámbito de la generación de energía eléctrica. En concreto, en este proyecto es de particular interés la energía eólica y las turbinas de viento. El uso de este tipo de energía crece cada vez más, y es que la energía eólica es una energía limpia, verde, segura y cada vez más barata.

Este crecimiento provoca que aumenten también el número de inspecciones de seguridad de las turbinas, pues también aumentan los fallos, los accidentes, las reparaciones, etc. Esto es necesario, pues sin un correcto mantenimiento cualquier tipo de estructura o elemento de grandes dimensiones disminuye su tiempo de vida muy rápido. En definitiva, al crecer el número de turbinas de viento, crece todo el entorno alrededor de estas. Esto supone un problema. Por un lado, aumentan los costes económicos relacionados con el mantenimiento, por el otro, realizar cualquier tarea de mantenimiento en una turbina eólica supone un grave peligro de seguridad para los operarios.

Una opción, es el uso de drones para realizar estas tareas de inspección. Estos drones pueden proporcionar mediante cámaras imágenes útiles para los operarios que realizan la inspección. Además, el uso de estos drones aumenta considerablemente la seguridad en este tipo de inspecciones. También, se mejora la productividad de la inspección al reducirse costes y disminuir el tiempo que se tarda en completar la misma.

Sería incluso una mejor idea si estos drones realizaran de forma autónoma la tarea de inspección. Los UAV (*Unmanned Aerial Vehicle*) permiten realizar una tarea de inspección de forma autónoma sin un piloto al mando de la operación (Sección 2.2). Por tanto, en suma, parece que la mejor opción es usar varios UAVs en un sistema multi-UAV para realizar estas inspecciones.

Al inicio de esta sección se han comentado las grandes oportunidades que ofrece la Realidad Virtual. Parece una idea razonable intentar combinar las ventajas que ofrece la Realidad Virtual con

las que ofrecen los sistemas multi-UAV en tareas de inspección. Este concepto de combinación de ambas tecnologías se muestra en la Figura 1.1.

Figura 1.1 Concepto de uso de Realidad Virtual y UAVs. Fuente: *Elaboración propia*.

Un ejemplo de este concepto se sintetiza en [1], donde se combinan la Realidad Virtual y los brazos robóticos. Se desarrolla una interfaz que permite la monitorización del brazo robótico y el control del mismo. Para ello, se consigue comunicar Unity, motor de Realidad Virtual, con ROS, un framework para el desarrollo de robots.

En [2], se presenta un método distribuido de planificación de trayectorias para sistemas multi-UAV en tareas de inspección de turbinas de viento (Sección 2.2.3). Este trabajo se desarrolla bajo el ámbito del proyecto europeo DURABLE [1] (Figura 1.2), cuyo objetivo es desarrollar tecnologías de inspección y mantenimiento de energías renovables.

Figura 1.2 Logo DURABLE. Fuente: *durableproject.eu*.

Lo interesante de este artículo, y que conecta con el concepto que se acaba de exponer, es que se sugiere como líneas de trabajo para el futuro la adaptación del HMI primitivo que opera la misión (ya desarrollado en el articulo) a un HMI en Realidad Aumentada/Virtual. La idea es desarrollar un *Virtual Cockpit* o Cabina Virtual que permita operar la tarea de inspección en un entorno virtual.

[1] durableproject.eu

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una Cabina de Realidad Virtual, propuesta en [2], para operar sistemas multi-UAV en tareas de inspección de turbinas de viento. Así, se busca producir una experiencia más inmersiva con las vistas de las cámaras y facilitar información relacionada con la misión de una forma más clara. Además, se quiere poder controlar los mismos parámetros y funcionalidades que en el HMI original. También, se añadirá un mapa 3D que permita tener una representación en el mundo virtual de lo que ocurre en el mundo real. La implementación final se ejecutará haciendo uso de unas gafas de Realidad Virtual.

Los objetivos planteados para lograr este propósito son los siguientes:

- Creación de un entorno de Realidad Virtual en Unity que satisfaga las necesidades de un operario que desea inspeccionar y controlar una turbina de viento con una flota de UAVs.
- Comunicación desde el entorno virtual en Unity con la simulación de la tarea de inspección en ROS para poder recibir las vistas de las cámaras de los UAVs y mostrar en una pantalla de status información de los UAVs y datos de la misión.
- Recreación del comportamiento real de la flota de UAVs mediante un mapa 3D, que permite al operario tener una perspectiva de lo que ocurre en el entorno real dentro del entorno virtual.
- Desarrollo de los controles necesarios para poder operar y controlar la Cabina Virtual. Estos controles se implementarán en unos mandos de Realidad Virtual.

2 Estado del arte

Antes de abordar la descripción de la metodología seguida para este proyecto, se va a realizar un estudio de los fundamentos en los que se enmarca todo el trabajo realizado.

En primer lugar, se tratará el tema de las energías renovables y cómo hoy en día se está impulsando el uso de estas por parte de diferentes agentes políticos, económicos y sociales. Después, se hablará del desarrollo y transformación que los UAV han experimentado a lo largo de los últimos años en tareas de inspección. Por último, se profundizará en una de las claves de este proyecto: las tecnologías de realidad virtual, aumentada y mixta, profundizando en la primera.

2.1 Fuentes de energía renovables

Las energías renovables han sido y serán uno de los tipos de energía clave para la sostenibilidad y desarrollo del planeta. Esto se debe a que son energías limpias, no contaminantes y que no se agotan. El uso de estas fuentes de energía se ha ido extendiendo a lo largo de los años y se planea que en el futuro sigan creciendo, ya que no producen gases de efecto invernadero y su coste se va encareciendo a medida que las tecnologías van evolucionando.

De acuerdo con Eurostat [1], en 2020 el porcentaje de energías de tipo renovable fue de media un 22,1 % en la Unión Europea, superando en un 2,1 % el objetivo para ese año, lo cual supone un aumento de más del doble de la media europea respecto al año 2004, donde el porcentaje fue de un 9,6 %. Algunos países como Suecia y Finlandia alcanzaron porcentajes del 60,1 % y 43,8 %, respectivamente. Uno de los objetivos de la Unión Europea, englobados dentro de la Agenda 2030, consiste en aumentar la media europea hasta el 40 %. A largo plazo, se planea aumentar ese porcentaje hasta el 55 % en 2050.

Algunas de las fuentes de energía renovables más utilizadas son la hidráulica, la solar y la eólica, entre otras. Es de especial interés para este proyecto la energía eólica, que se transforma en energía eléctrica gracias a los aerogeneradores, cuyas palas son movidas por el viento.

La energía eólica es una de las energías renovables más importantes dentro de la Unión Europea y a nivel global. Es una fuente de energía barata, eficiente y sostenible. De hecho, en 2020 representó el 36 % del total de energías renovables en la Unión Europea (Figura 2.1), siendo la mayor en porcentaje de estas. Además, según WindEurope [2], la capacidad de potencia eólica en Europa en 2021 era de 236 GW (Figura 2.2), lo que se traduce en más del doble de capacidad desde 2012.

Sin duda, conforme a los datos anteriores, las energías renovables y en concreto la energía eólica, desempeñan un papel fundamental en la producción de electricidad. Es una fuente de energía en auge, verde, segura y de bajo impacto, con grandes ventajas frente a sus desventajas. Se postula

[1] ec.europa.eu/eurostat

[2] windeurope.org/intelligence-platform

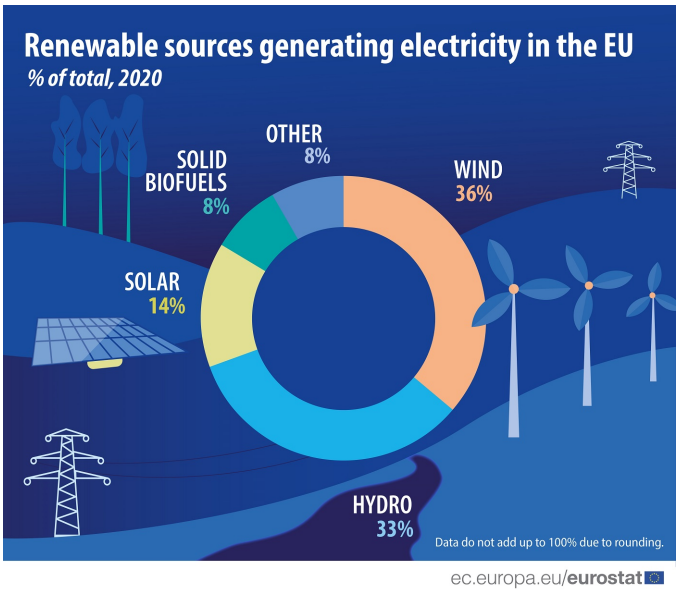

Figura 2.1 Energías renovables por tipo en la Unión Europea (en porcentaje), 2020. Fuente: *Eurostat*.

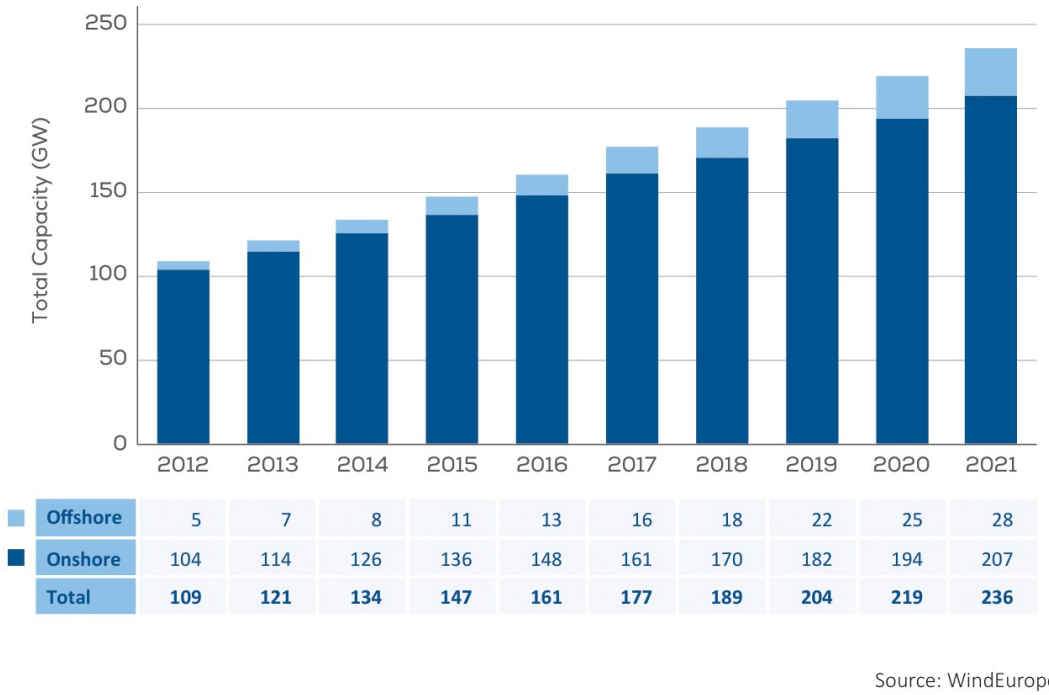

Figura 2.2 Crecimiento de la capacidad total de energía eólica en Europa (en GW), 2012-2021. Fuente: *WindEurope*.

como una tecnología clave para cumplir objetivos climáticos y como futuro modelo energético a nivel global.

Es por ello que diferentes organismos políticos, económicos y sociales están impulsando el uso de energías renovables como la eólica. El Acuerdo de París de 2015 [3], la COP26 [4], la Agenda 2030 [5] o los objetivos propuestos por la Unión Europea son algunos ejemplos.

[3] unfccc.int
[4] ukcop26.org/energy
[5] ec.europa.eu

En particular, como ya se mencionó en la Introducción (Capítulo 1), este trabajo se ha realizado en el ámbito del proyecto europeo DURABLE [6], cuyo objetivo es desarrollar nuevas tecnologías que favorezcan a la inspección y mantenimiento de fuentes de energía renovables en la región atlántica, esencialmente en paneles solares y turbinas de viento.

2.2 UAVs

Aquí se tratará la evolución y aplicaciones de los UAVs. En particular, se hará hincapié en las aplicaciones de los UAVs en tareas de inspección y las ventajas que estos aportan a las mismas.

2.2.1 Evolución de los UAVs

El desarrollo de los primeros UAVs estuvo intrínsecamente ligado a aplicaciones de uso militar [3]. En Julio de 1849, las fuerzas austriacas que intentaban ocupar Venecia lanzaron 200 balones incendiarios cargados con bombas (Figura 2.3). Esto es considerado como el primer registro que se tiene del uso de UAVs. Más tarde, durante la Primera y Segunda Guerra Mundial, se introdujo el concepto de radio control y los UAVs evolucionaron rápidamente hacia aplicaciones más diversas, fiables y precisas.

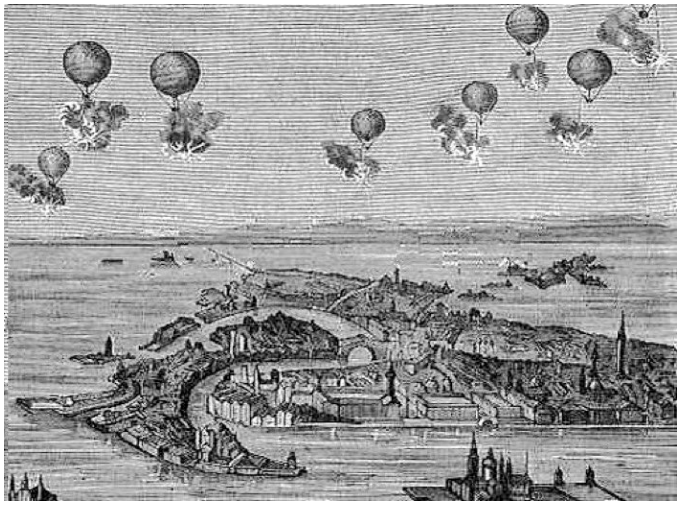

Figura 2.3 Ataque austriaco con balones incendiarios sobre la ciudad de Viena, 1849. Fuente: *historytoday.com*.

Desde entonces, el uso de los UAVs ha ido creciendo al mismo tiempo que sus tecnologías, en aplicaciones militares [4] pero también en aplicaciones de aviación general y de consumo. En los últimos años, los UAVs están experimentando su momento de mayor auge y la sociedad está empezando a descubrir las ventajas de estos. En parte, esto es gracias al progreso de tecnologías más inteligentes, a la implementación de cámaras, a la reducción de tamaño y a la mejora de la autonomía de las baterías. El *quadrotor* (Figura 2.4) es el claro ejemplo de este aumento de popularidad de los UAVs ya que, hoy día, es usado en muchas aplicaciones aéreas civiles.

2.2.2 Aplicaciones generales

Como se mencionaba en la Sección 2.2.1, los UAVs se han usado desde siempre en operaciones militares. Recientemente, las aplicaciones civiles donde se emplean UAVs han crecido en gran medida [5]. Por ejemplo, son usados como ayudante para tareas de vigilancia y seguridad [6], en

[6] durableproject.eu

Figura 2.4 Quadrotor DJI Mini 3 Pro. Fuente: *DJI*.

entrega de carga [7], acceso a lugares de difícil exploración [8], búsqueda y rescate [9], etc. Esto es solo una pequeña muestra del uso que se esta haciendo de los UAVs. A continuación, se exponen una serie de ejemplos de aplicaciones reales (Figura 2.5).

Otra aplicación de interés en la que se está apostando mucho por los UAVs es en la filmación aérea, aunque aún es prematura en el sentido de que tiene flaquezas como cubrir como, por ejemplo, la iluminación. En [10], se muestra un método para sistema multi-UAV con iluminación distribuida. Es común realizar grabaciones en espacios con abundante luz, pero resulta menos conveniente realizar estas grabaciones en entornos con escasa iluminación. Aquí se plantea una solución que permite iluminar desde varias perspectivas para mejorar la calidad de la iluminación. Se utiliza un UAV líder que graba el objetivo y dos seguidores que iluminan al objetivo desde posiciones distintas. Los resultados se demuestran tanto en simulación como en un escenario real.

Figura 2.5 Ejemplos de aplicaciones de UAVs. Fuente: *Elaboración propia*.

Respecto a tareas de vigilancia y seguridad, por ejemplo, abunda el empleo de UAVs para detección de incendios. En [11], se deja ver la eficacia de los UAVs en este ámbito dada la dificultad de acceso a los incendios, y se presenta un algoritmo de planificación de trayectorias que utiliza imágenes de cámaras infrarrojas. Este algoritmo es muy útil pues tener datos actualizados y en tiempo real de la evolución de un incendio es clave para mitigar los daños. Se presentan los resultados en simulación utilizando el modelo EMBYR que emula incendios forestales. En concreto, se utiliza un sistema multi-UAV, de baja altitud y poca autonomía, pues los datos combinados de varios UAVs mejoran significativamente la información que se obtiene del crecimiento y evolución del incendio.

La principal fuente de información de utilidad que se obtiene de un UAV es la visual, y es por ello que suelen tener acoplada una cámara (RGB, térmica, infrarroja, etc.) que, según el propósito, la

tendrá colocada en un sitio u otro. Hablando del potencial que puede tener la incorporación de esta junto con cierta algorítmica, por ejemplo, en [12] se expone un método que utiliza tratamiento de imágenes en varias etapas. Utiliza filtrado de mediana, conversión de color, segmentación de Otsu, operaciones morfológicas y contador de manchas. Tras este tratamiento de imágenes, se utiliza el canal "a" para marcar el fuego en la imagen. El canal "a" contiene los tonos rojos y verdes de la imagen. En los resultados se demuestra que el canal "a" es el que aporta resultados más eficaces. Por supuesto, estas imágenes son obtenidas mediante cámaras colocadas en UAVs al igual que el ejemplo anterior.

En [13], se propone un modelo de *deep reinforcement learning* que revisa la planificación del movimiento de un drone que graba, haciendo selecciones del tipo de toma en cada momento. Estas decisiones provienen del entrenamiento del modelo con experiencia previa. Se consigue un resultado estético y similar al obtenido por un operador humano.

Otra aplicación de interés es la agricultura. En [14], se diseña un drone con sistema pulverizador de bajo coste. La idea es reducir las muertes causadas por pesticidas haciendo uso de drones en lugar de las técnicas tradicionales para hacer uso de estos. El manejo de los UAVs se realiza de manera semiautónoma y se deja para el futuro el uso de UAVs completamente autónomos. Se utiliza un software planificador de misiones al que se le proporciona la ruta a seguir previamente. Los resultados son probados en experimentos reales y la trayectoria real se ajusta prácticamente a los waypoints de la ruta predefinida. Enlazando con el ámbito sensorial y de comunicaciones, en [15] se muestra una estructura de colaboración entre un UAV y una red de sensores WSN *(Wireless Sensor Networks)*, en este caso para agricultura de precisión. El objetivo principal es la integración de la información recogida por los UAVs y los sensores en tierra utilizando IoT *(Internet of Things)*. Se realizan experimentos en un entorno real donde se hacen notar las ventajas de este método: recopilación y procesamiento de datos en tiempo real y diseño de trayectorias optimizadas para la recogida de datos de los UAVs.

Finalmente, como se verá en la Sección 2.2.3, una de las aplicaciones más recientes en la que los UAV son de gran utilidad es la de las tareas de inspección. Esta aplicación es, además, especialmente relevante para este trabajo pues la Cabina Virtual ha sido diseñada específicamente para operar un sistema multi-UAV en una tarea de inspección de turbinas de viento.

2.2.3 UAVs en tareas de inspección

Ya se ha visto que los UAVs poseen características que los hacen ser una buena opción para ciertas aplicaciones (Sección 2.2.2). Los UAVs usados en tareas de inspección permiten realizar inspecciones en lugares peligrosos y de difícil acceso como por ejemplo una turbina de viento, una torre de antenas o un puente. En inspecciones visuales el uso de varios UAVs permite obtener varios puntos de vista de un mismo punto objetivo. Se consigue una mayor eficacia y precisión respecto a las tareas de inspección tradicionales, además, se aumenta la seguridad de los operarios y se reducen costes significativamente. Por supuesto, también se reduce el tiempo total de operación y se facilita en conjunto la tarea de inspección.

Aun así, esto plantea algunos retos que deben ser abordados. Por ejemplo, en [16], se realiza un análisis del uso de UAVs en la inspección interna y externa de las palas de las turbinas de viento. En cuanto a la inspección externa, se evitan los trabajos verticales y el uso de andamios. Sin embargo, las inspecciones internas suponen un mayor reto debido al reducido espacio, la falta de iluminación y los elementos estructurales internos que pueden dificultar el movimiento. El peso, el tiempo de vuelo y la estrategia de vuelo limitan las capacidades de los UAVs en la inspección interna.

Como se vio en la Sección 2.1, las energías renovables juegan un papel fundamental en la actualidad. La energía solar es un activo muy importante en la generación de energía eléctrica y los paneles solares son los encargados de llevar esto a cabo. En [17], se presenta un sistema que es capaz de supervisar las operaciones de un sistema de placas solares. Cuenta con una cámara CCD *(Charge-Coupled Device)* que recibe por separado cada rango del espectro de colores y una cámara

térmica montadas sobre un drone. El objetivo es detectar fallas en los paneles solares con un sistema más seguro y eficiente. Como resultado, el sistema es capaz de analizar vídeos de las dos cámaras para localizar fallos internos y externos en los paneles fotovoltaicos y, a a la vez, da información de la localización del fallo en el panel. En la Figura 2.6 se muestran imágenes obtenidas por la cámara térmica. Puede apreciarse en color amarillo la localización de los "puntos calientes", es decir, los fallos internos de los paneles solares.

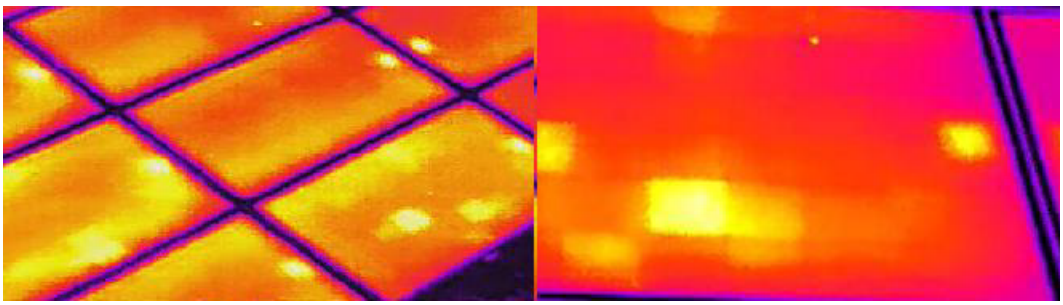

Figura 2.6 Puntos calientes en los paneles solares. Fuente: [17].

En [18], se utiliza un sistema multi-UAV con una arquitectura multicapa para ayudar en tareas de inspección de líneas eléctricas. El software está dividido en tres capas que permite modularidad: el gestor de tareas, los sistemas de inspección y los sistemas de seguridad. Con esta aplicación se quiere dar apoyo a la inspección de líneas eléctricas y al mismo tiempo brindar seguridad al operario que está realizando la tarea. Los resultados se muestran en simulaciones realistas en Gazebo.

Por último, en lo que concierne a tareas de inspección, cabe destacar el siguiente artículo [2]. Aquí se propone un método distribuido de planificación de trayectorias para sistemas multi-UAV en tareas de inspección de turbinas de viento. El sistema de UAVs cuenta con un UAV líder que sigue una serie de waypoints, generando así trayectorias para alcanzar dichos puntos; y otros UAVs seguidores que replican las trayectorias generadas por el líder. Lo característico de este sistema, es que el líder es quien sigue los waypoints y no los seguidores, estos solo mantienen la posición relativa respecto al líder. Al igual que en [11], el uso de varios UAVs permite obtener más variedad de información, en este caso, imágenes.

Esta formación de UAVs puede ser operada desde tierra por un operario gracias a un HMI *(Human Machine Interface)*, como se puede observar en la Figura 2.7. El operario puede manipular el sistema en su conjunto: puede modificar parámetros como la distancia a los puntos de inspección y el ángulo relativo, aterrizar y despegar los UAV, etc.

En el articulo, se sugiere como trabajo futuro la creación de un HMI en Realidad Aumentada/Virtual que combine las imágenes de los tres UAVs para producir una experiencia más inmersiva y facilitar información relacionada con la misión.

Es aquí, como se comentó en la motivación de la Introducción (Sección 1.1), donde nace la idea de realizar este Trabajo Fin de Grado, es decir, realizar una Cabina de Realidad Virtual usando el motor de juego Unity para operar el sistema multi-UAV. En esencia, se pretende integrar el HMI preliminar en un entorno virtual, donde las vistas de las cámaras proporcionen una sensación de inmersividad en la escena y donde los datos de la misión sean mostrados de una forma más visual y comprensible. Además, se busca poder modificar los mismos parámetros que en el HMI original. También, se añadirá un mapa 3D que permita tener una representación en el mundo virtual de lo que ocurre en el escenario real.

Figura 2.7 Idea preliminar del HMI *(Human Machine Interface)*. Fuente: [2].

2.3 Realidad Aumentada, Virtual y Mixta

Si en la Sección 2.2.1 se mencionaba que los UAVs se encuentran en su momento de mayor auge, la Realidad Virtual sigue un camino similar. La Realidad Aumentada, Virtual y Mixta están evolucionando constantemente como se puede apreciar en algunos ejemplos que se verán más adelante pero, sin duda, un hito que muy recientemente ha marcado el futuro de la Realidad Virtual es el cambio de nombre de la empresa tecnológica Facebook a Meta (Figura 2.8), el pasado 28 de Octubre de 2021 [7].

El objetivo principal de este cambio de nombre es reflejar el giro de la empresa hacia el *"Metaverso"*, donde aspiran a conectar a la gente de nuevas maneras haciendo uso de la Realidad Aumentada y Virtual. Además, Meta dispone de la gama Meta Quest (anteriormente Oculus Quest) (Figura 2.9) de gafas de Realidad Virtual para hacer esto posible (Sección 3.3).

Figura 2.8 Logo de Meta. Fuente: *meta.com*.

Cabe destacar que se conoce como Realidad Extendida al conjunto de la Realidad Aumentada, Virtual y Mixta. A continuación, se realiza una breve introducción de cada tecnología y se presentan trabajos relacionados con las mismas.

[7] about.fb.com

Figura 2.9 Gafas de Realidad Virtual Meta Quest. Fuente: *Meta Quest*.

2.3.1 Realidad Aumentada

En la Realidad Aumentada, AR o *Augmented Reality*, se superponen capas de información sobre la visión del mundo real mediante gráficos digitales en tiempo real. Es decir, se parte de la base del mundo real y encima de este se añade información virtual. Esta tecnología se caracteriza por no proporcionar interactividad con los gráficos superpuestos.

El muestreo de suelo es realmente necesario en la agricultura de precisión. En [19], se presenta una aplicación que integra la Realidad Aumentada y las imágenes tomadas por drones para determinar automáticamente donde realizar las muestras del suelo. Se consigue guiar al usuario hasta el punto donde se tiene que realizar la muestra usando gafas de Realidad Aumentada. Gracias a estas gafas, puede ver información relevante que le ayude en el proceso de toma de muestras.

En [20], se crea un sistema de Realidad Aumentada que busca unificar toda la información relacionada con la ruta a seguir por un UAV para facilitar su comprensión. Esta herramienta es clave para ayudar en la orientación de la ruta y la identificación del objetivo del UAV.

2.3.2 Realidad Virtual

La Realidad Virtual, VR o *Virtual Reality*, consiste en la creación de mundos totalmente virtuales. En este caso no hay ninguna interacción con el mundo real, esto permite una mayor inmersividad en el mundo virtual. Aunque esta tecnología siempre ha estado enfocada al mundo de los videojuegos, cada vez son más las aplicaciones en las que se utiliza la Realidad Virtual. De las tres, esta es la tecnología usada en este proyecto. El uso de la Realidad Virtual permite al operario sumergirse en la Cabina Virtual y prestar atención solo a aquello relativo a la misión de inspección.

La Realidad Virtual puede usarse para mejorar el conocimiento de la situación de los pilotos de helicópteros. En [21], se introduce un *Virtual Cockpit* (Figura 2.10) que permite al piloto disponer de más información del exterior ya que a veces las condiciones de visibilidad son bajas. A través de este sistema puede mostrarse al usuario información como datos sobre obstáculos, terreno o tráfico, parámetros de vuelo e imágenes de los sensores de visión.

En [22], se desarrolla una interfaz llamada WareVR que permite operar un sistema robótico formado por un UAV y un UGV *(Unmanned Ground Vehicle)* para gestión de inventarios de un almacén. En esta interfaz se muestran imágenes en tiempo real del almacén y se pueden realizar diferentes inventarios, supervisar el proceso automático de inventario y operar el drone de forma manual (Figura 2.11).

2.3.3 Realidad Mixta

La Realidad Mixta, MR o *Mixed Reality*, combina Realidad Aumentada y Virtual. En este caso ya no solo se tiene un mundo completamente virtual o solo se superpone información sobre el mundo real, sino que se mezcla el mundo real y virtual. Esto produce una sensación más inmersiva que la Realidad Virtual. La Realidad Mixta ofrece más posibilidades: interaccionar con objetos del mundo

Figura 2.10 Piloto usando las gafas Oculus Rift en el Virtual Cockpit. Fuente: [21].

Figura 2.11 Interfaz de WareVR. Fuente: [22].

real dentro de un mundo virtual, mostrar elementos virtuales en el mundo real o permanecer en un mundo puramente virtual. Gracias a esto se pueden generar nuevos espacios en los que se mezclan elementos y personas tanto del mundo real como del mundo virtual. En este caso, la principal

diferencia con la Realidad Aumentada es que esta no ofrece interacción con los elementos virtuales mientras que la Realidad Mixta sí.

En [23], se implementa un sistema de fusión sensorial para inspección de incendios donde se incorporan datos de una IMU (*Inertial Measurement Unit*), una cámara, un algoritmo de detección de incendios y un UAV. Además, se crea un entorno de Realidad Mixta para probar el algoritmo, y así, comunicar el mundo real y el mundo virtual mediante una comunicación en tiempo real. El objetivo es ampliar el abanico de casos donde se pueden realizar pruebas del algoritmo de detección de incendios.

En [24], se realiza algo similar a [23] y se presenta un método para inspección de estructuras basado en Realidad Mixta. En entornos de grandes dimensiones la localización de las gafas de Realidad Mixta es complicada, por ello, se utiliza un método de auto localización que utiliza el sensor de profundidad y la cámara parar generar nubes de puntos de pequeñas dimensiones en lugar de una pero de mayor tamaño.

3 Tecnologías utilizadas

Ahora se van a explicar las tecnologías utilizadas para este proyecto y para qué sirve cada una de ellas. Uno de los objetivos a cumplir (Sección 1.2) es el de realizar una conexión entre Unity y ROS, por tanto, estas tecnologías junto con las gafas de Realidad Virtual utilizadas serán claves en este capítulo.

3.1 Unity

Unity (Figura 3.1) es un motor de juego perteneciente a Unity Technologies, líder en la industria del desarrollo de videojuegos y entornos gráficos. Es un motor multiplataforma que soporta la gran mayoría de sistemas operativos, consolas, dispositivos móviles y de Realidad Extendida (Sección 2.3). Esto último es uno de los motivos por los que se ha escogido Unity como herramienta de desarrollo para la Cabina de Realidad Virtual. Existen otras opciones en el mercado, pero Unity es uno de los motores gráficos más usados por su robustez, versatilidad y sencillez [25]. Además, en lo relativo a la Realidad Extendida, Unity destaca especialmente frente a sus competidores.

Figura 3.1 Logo Unity. Fuente: *unity.com*.

Unity utiliza el lenguaje de programación C# y como se ha mencionado anteriormente, es un motor multiplataforma. Esto permite desarrollar una aplicación gráfica y poder utilizarla en una amplia gama de dispositivos sin modificarla. Unity no es por sí solo un motor dedicado al desarrollo de Realidad Virtual, pero utilizando ciertas herramientas se convierte en unas de las mejores aplicaciones para ello. La documentación oficial de Unity [1] cuenta con un apartado específico dedicado a la Realidad Extendida.

[1] docs.unity3d.com/XR

En este proyecto se utiliza la versión de Unity 2020.3.32f1 LTS [2] y las gafas de Realidad Virtual Oculus Quest (Sección 3.3), por lo tanto, se hace uso del paquete *Oculus Integration* [3], un SDK *(Software Development Kit)* que contiene herramientas de todo tipo para facilitar el desarrollo de aplicaciones de Realidad Virtual en dispostivos de Oculus en Unity.

A continuación, se hará una breve introducción al entorno de Unity. Se puede dividir la interfaz de Unity en cuatro partes principales (Figura 3.2):

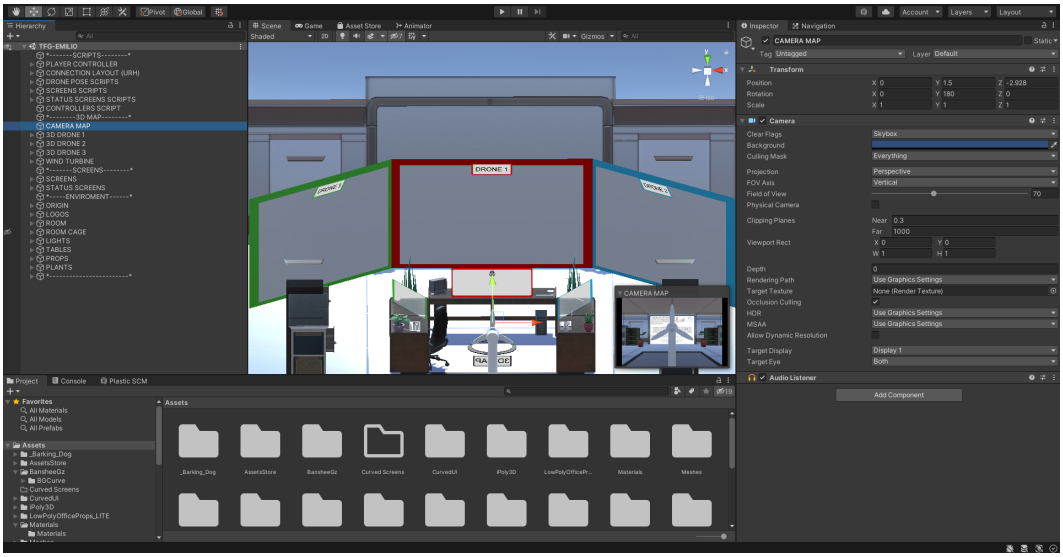

Figura 3.2 Interfaz de Unity, Versión 2020.3. Fuente: Elaboración propia.

- Jerarquía *(Hierachy)*: La ventana más a la izquierda. Es una representación jerárquica de la estructura de todos los *GameObject* de la escena de Unity. Un *GameObject* es el objeto más importante en Unity. Básicamente es un contendedor de objetos. En función de lo que contenga, el *GameObject* tendrá unas características u otras.
- Vista Escena/Juego *(Scene/Game View)*: La ventana del centro. La Vista Escena permite moverse por la escena y editarla. La Vista Juego simula lo que se verá en el renderizado final de la escena.
- Proyecto *(Project Window)*: La ventana de abajo. Es el explorador de archivos de Unity. Aquí aparecen todos los *Assets* del proyecto. Un *Asset* es un elemento de Unity y puede ser un modelo 3D, audio, una imagen, etc.
- Inspector *(Inspector)*: La ventana más a la derecha. La ventana Inspector permite ver y editar las propiedades de los *GameObject*.

Algunos conceptos clave de Unity son:

- *Textures*: Son la base de la superficie de un objeto. Junto con los Materials y Shaders establecen el resultado final de la superficie.
- *Materials*: Establecen cómo se renderiza la superficie de un objeto.
- *Shaders*: Calculan el color exacto de cada píxel de una superficie teniendo en cuenta parámetros como la iluminación.
- *Meshes*: Definen la forma de un objeto.

[2] unity3d.com/es/unity/qa/lts-releases
[3] developer.oculus.com/downloads

3.2 ROS

ROS [4] *(Robot Operating System)* (Figura 3.3) fue creado en 2007 en el Stanford Artificial Intelligence Lab. Es un conjunto de paquetes, librerías y herramientas (no un sistema operativo como tal) que facilita el desarrollo de software de robots. ROS es el software utilizado para simular la tarea de inspección.

Figura 3.3 Logo ROS. Fuente: *ros.org*.

Es un framework de desarrollo que aporta servicios típicos de sistemas operativos como la abstracción de hardware, el control de dispositivos de bajo nivel, el paso de mensajes entre procesos y mantenimiento de paquetes. Permite establecer y controlar la comunicación entre los distintos procesos de un robot.

ROS es el framework más usado en aplicaciones de robótica porque posee características únicas como la flexibilidad, modularidad, escalabilidad. Además, es un software libre y de código abierto, cuenta con una gran comunidad de desarrollo y permite la integración con simuladores como Gazebo [5].

ROS posee una arquitectura de grafos basada en los siguientes elementos principales:

- Nodos: Módulos que realizan una tarea concreta. Es donde se reciben, procesan y emiten datos.
- Topics: Canales para compartir mensajes entre diferentes nodos.
- Mensajes: Datos que se mandan a través de los topics.
- Servicios: Otro método de comunicación entre nodos. Permiten que un nodo envíe una solicitud a otro y recibir una respuesta de manera síncrona.

3.3 Oculus Quest

Las aplicaciones en Realidad Virtual necesitan de un dispositivo especial para poder ser ejecutadas. Existen diversas opciones, todas con características similares, pero a la vez diferentes. Para este proyecto se han utilizado las gafas de Realidad Virtual Oculus Quest [6] (Figura 2.9 y 3.4) de Meta.

Figura 3.4 Logo Oculus. Fuente: *oculus.com*.

[4] ros.org
[5] gazebosim.org
[6] store.facebook.com/quest/products/quest

En la documentación oficial de Oculus [7] puede encontrarse todo lo necesario para empezar a desarrollar una aplicación de Realidad Virtual en Oculus Quest, en este caso, con Unity.

Estas gafas cuentan con un panel OLED de 1600 x 1440 píxeles con una tasa de refresco de 72 Hz y audio posicional 3D integrado en las gafas. Poseen 4 GB de memoria y seis grados de libertad, lo que permite realizar seguimiento de los movimientos de la cabeza y cuerpo sin sensores externos. La CPU es un Qualcomm® Snapdragon 835 y la GPU un Qualcomm® Adreno™ 540 GPU.

Además de las gafas, se necesitan mandos o controladores para interactuar con el mundo virtual. En este caso, se usan los mandos Oculus Touch. En la Figura 3.5 se pueden apreciar los distintos botones, gatillos y joysticks que contienen los mandos.

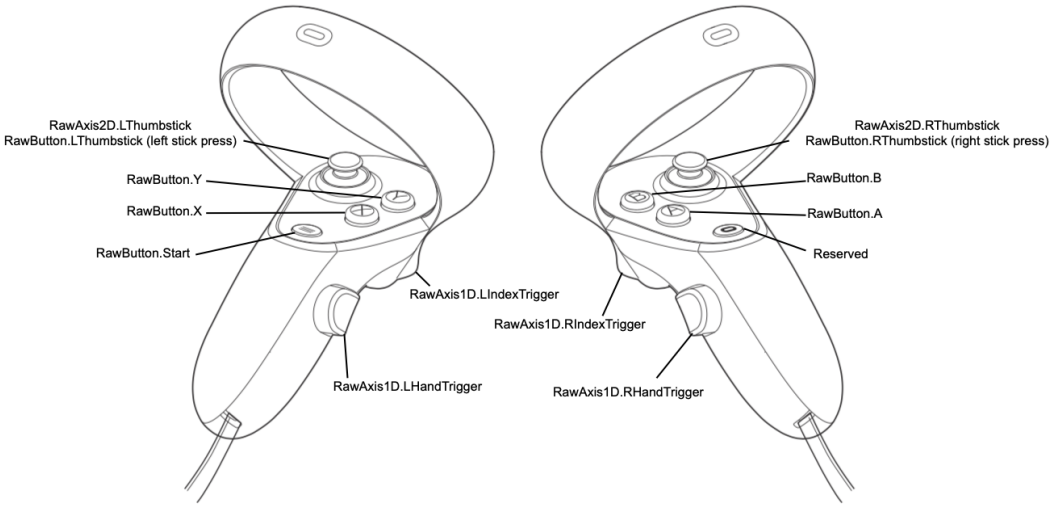

Figura 3.5 Controladores Touch de Oculus Quest. Fuente: *developer.oculus.com/unity-ovrinput*.

A continuación, se presenta la Tabla 3.1 que muestra con más detalle cada uno de estos elementos:

Tabla 3.1 Botones, gatillos y joysticks de los controles Oculus Quest.

	Mando izquierdo	Mando derecho
Botones	Botón X e Y. Botón Joystick izquierdo. Botón Start	Botón A y B. Botón Joystick derecho
Gatillos	Lateral y trasero	Lateral y trasero
Joysticks	Joystick izquierdo	Joystick derecho

En la Sección 4.4.3, se explica con más profundidad la funcionalidad de cada elemento de los controles.

[7] developer.oculus.com/documentation

4 Metodología

En este capítulo se van a desarrollar los procedimientos realizados para llevar a cabo este proyecto. Se describe el sistema en su conjunto (Sección 4.1) y, posteriormente, la conexión realizada entre Unity (Sección 3.1) y ROS (Sección 3.2) en la Sección 4.2. Después, se explica la simulación que se ejecuta en ROS (Sección 4.3) y, finalmente, el entorno de la aplicación de Realidad Virtual en Unity (Sección 4.4).

El código del proyecto completo de este Trabajo Fin de Grado puede encontrarse en el enlace al repositorio a pie de página [1], junto con un vídeo ilustrativo de los resultados del mismo.

4.1 Arquitectura del sistema

En este proyecto, como se comentó en el Capítulo 1, se desarrolla una Cabina en Realidad Virtual. El objetivo principal de esta aplicación es operar un sistema multi-UAV en una tarea de inspección de turbinas de viento. La tarea de inspección se simula en ROS mientras que el entorno virtual se crea en Unity.

El sistema está separado en dos partes distintas: una para la tarea de inspección (ROS) y otra para la Cabina Virtual (Unity/Oculus Quest). El sistema podría funcionar en un único PC, pero debería tener unas prestaciones muy altas debido a la alta carga computacional del sistema completo.

En la Figura 4.1 se observan claramente la dos partes de la arquitectura en las que se divide el proyecto. Es necesario crear una conexión entre ambas partes, para lo cual se usa ROS#, que hace uso de *rosbridge_suite* y crea una conexión mediante WebSocket. Esto se verá en detalle en la Sección 4.2.

En la parte derecha, se observa el PC con Ubuntu 20 y ROS Noetic. En este PC se simula la misión de inspección haciendo uso de ROS y del simulador Gazebo (Sección 4.3).

En la parte de la izquierda, aparece el PC con Windows 10 y las gafas de Realidad Virtual Oculus Quest. Se enumeran las cinco partes principales en las que se organiza la Cabina Virtual: la simulación del entorno, la vista de las cámaras de los UAV, la mesa de status y control, los controles para operar la misión y el mapa 3D, que permite visualizar dentro del mundo virtual lo que ocurre en el mundo real. En el PC se simula el entorno virtual con Unity (Sección 4.4). Cabe destacar que, esta misma aplicación desarrollada en Unity con todas sus características y propiedades, es posteriormente ejecutada en las gafas de Realidad Virtual Oculus Quest (Sección 3.3).

[1] Repositorio del proyecto

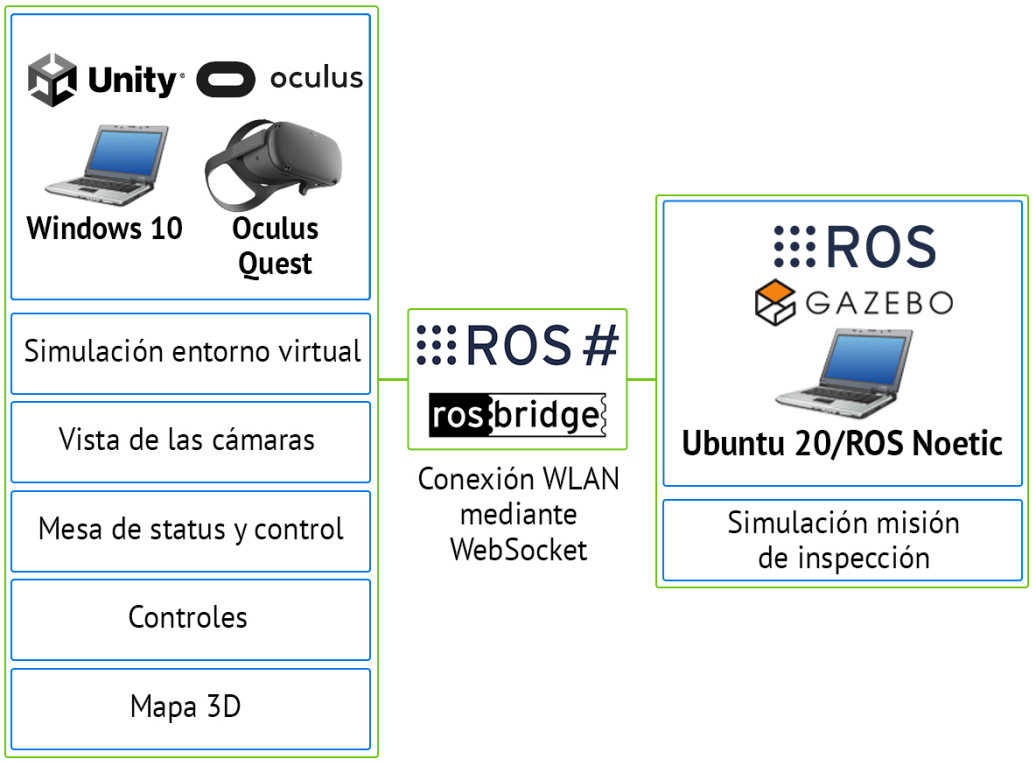

Figura 4.1 Arquitectura del sistema. Fuente: Elaboración propia.

4.2 Comunicación entre Unity y ROS

Para poder realizar la conexión entre Unity y ROS se utiliza principalmente ROS# o ROS Sharp [2], desarrollado por Siemens. Además, se utiliza Unity Robotics Hub [3], desarrollado por Unity Technologies.

4.2.1 Unity Robotics Hub

Unity Robotics Hub es un repositorio de documentación, herramientas, recursos y tutoriales para la simulación robótica en Unity. Cuenta con funcionalidades prácticamente idénticas a las de ROS#, sin embargo, en este proyecto se utiliza ROS# para establecer la comunicación entre Unity y ROS como se verá a continuación.

Aun así, de Unity Robotics Hub se utiliza el paquete de Visualizaciones *(Visualizations)* [4] para poder visualizar la información entrante y saliente de los topics en Unity (Figura 4.2).

Unity Robotics Hub se divide en dos partes, una para el lado de ROS y otra para el de Unity:

- *ROS TCP Endpoint* [5]: Nodo de ROS en el que se abre un puerto (servidor) para poder establecer comunicación con Unity.
- *ROS TCP Connector* [6]: Paquete de Unity que permite conectarse como cliente a ROS a través de una misma red para poder interaccionar con él.

[2] github.com/siemens/ros-sharp
[3] github.com/Unity-Technologies/Unity-Robotics-Hub
[4] github.com/Unity-Technologies/Visualizations
[5] github.com/Unity-Technologies/ROS-TCP-Endpoint
[6] github.com/Unity-Technologies/ROS-TCP-Connector

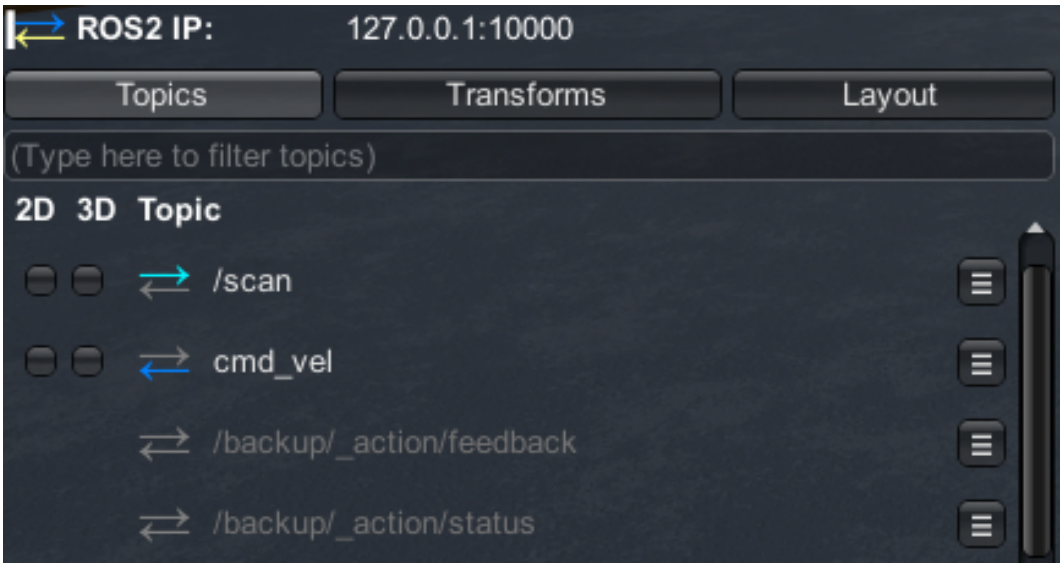

Figura 4.2 HUD de Unity Robotics Hub. Fuente: *github.com/Unity-Technologies*.

Pese a todo, como se ha comentado antes, esto no se utiliza en la implementación final, pero sí algo muy parecido de la librería ROS#. Se utiliza ROS# porque ofrece mayor flexibilidad para tratar la información y es una herramienta más completa. Además, Unity Robotics Hub está desarrollado a partir de ROS# [7], y no está tan depurado como este último.

4.2.2 ROS#

ROS# (Figura 4.3) es un conjunto de herramientas y bibliotecas de software en C# para comunicarse con ROS desde Unity. Cuenta con paquetes para ROS y un proyecto de Unity. Permite crear, haciendo uso de *rosbridge_suite* [8], una conexión LAN/WLAN mediante WebSocket entre Unity y ROS, conectados a una misma red y con sus correspondientes IP.

Figura 4.3 Logo ROS#. Fuente: *github.com/siemens/ros-sharp*.

En cuanto a la parte de ROS contiene, entre otros, el paquete *file_server*. En este paquete se encuentra *ros_sharp_communication.launch*, que se ejecuta para configurar la conexión. A su vez, esto lanza *rosbridge_server*. Así, se crea una conexión WebSocket bidireccional y persistente en el puerto 9090 y con la IP del PC que ejecuta ROS.

En lo referente a Unity, se hace uso del componente *RosConnector* que se añade mediante el paquete de ROS# del Asset Store de Unity [9]. Este componente establece la comunicación con ROS, basta con configurar la IP del PC que ejecutará ROS y el puerto correspondientes.

Es importante hacer uso de *MessageGeneration* en Unity para generar el código en C# de los mensajes de ROS definidos por el usuario, en caso contrario, ROS# no podrá trabajar con estos mensajes.

[7] github.com/siemens/ros-sharp/wiki
[8] wiki.ros.org/rosbridge_suite
[9] assetstore.unity.com

En conclusión, como se refleja en la Figura 4.4, se crea una conexión entre Unity y ROS haciendo uso de ROS#. En esta conexión, *rosbridge_server* se ejecuta en ROS y *RosBridgeClient* en Unity. Como resultado, se obtiene una comunicación rápida y fiable entre las dos partes del sistema.

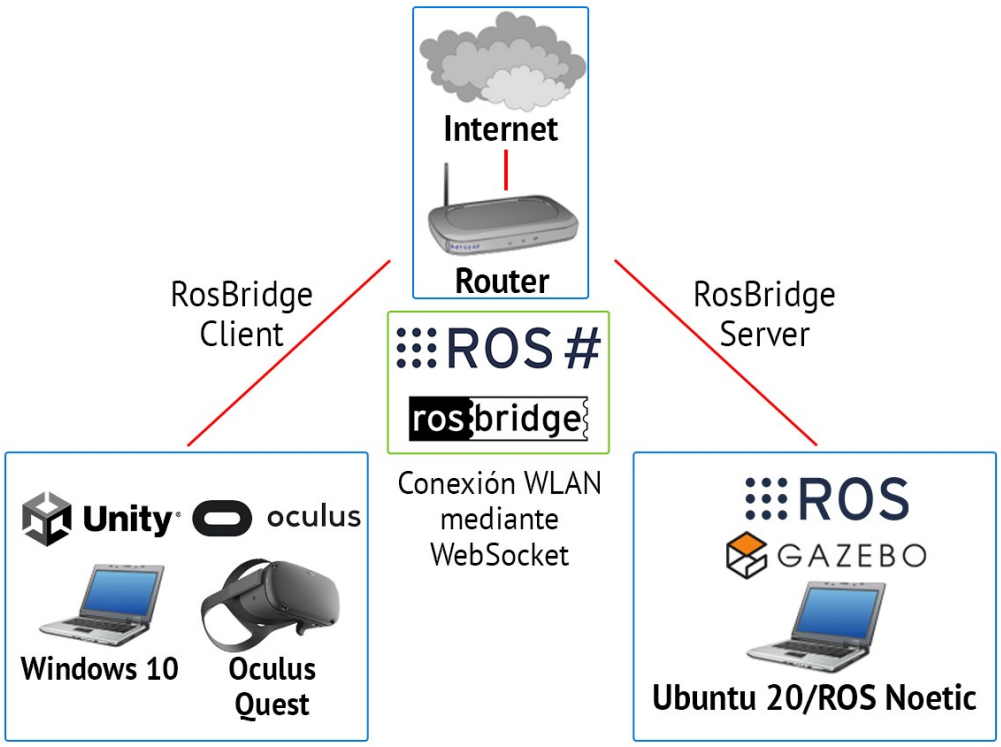

Figura 4.4 Diagrama de comunicaciones del sistema. Fuente: Elaboración propia.

4.3 Tarea de inspección en ROS/Gazebo

La aplicación de Realidad Virtual se desarrolla para operar una misión de inspección, por tanto, esta misión debe ser simulada para poder interactuar con la Cabina Virtual.

Como se mencionaba en el Capítulo 1, este proyecto surge del artículo [2], donde el autor presenta un método distribuido de planificación de trayectorias para sistemas multi-UAV en tareas de inspección de turbinas de viento.

La simulación que se desarrolla en [2] es la seleccionada para interaccionar con la Cabina Virtual. Esta simulación puede encontrarse en un repositorio online [10]. En la Figura 4.5 se observa el mundo que se ha creado, donde puede apreciarse una turbina de viento y varios UAVs. La simulación se ejecuta desde ROS Noetic en Ubuntu 20 y se usa el simulador Gazebo [11].

La simulación consiste en una formación de tres UAVs: uno líder y dos seguidores. Cuentan con una arquitectura líder-seguidor, es decir, los seguidores dependen del UAV líder (Figura 4.6). Las trayectorias generadas para el líder son circulares y se replican para los seguidores, haciendo una rotación de cierto ángulo de formación respecto al eje Z y centrado en la turbina de viento. Estas trayectorias se generan a partir de waypoints que pueden ser añadidos manualmente.

Hay parámetros que pueden ser modificados dentro de la simulación, tales como la distancia de inspección, el ángulo de formación o el tiempo de órbita. Los UAVs se pueden despegar y aterrizar, además, se puede iniciar o parar la misión.

[10] github.com/grvcTeam/inspection_trajectory_planning
[11] gazebosim.org

Figura 4.5 Mundo en Gazebo con la formación de UAVs ejecutando la misión de inspección de turbinas de viento. Fuente: [2].

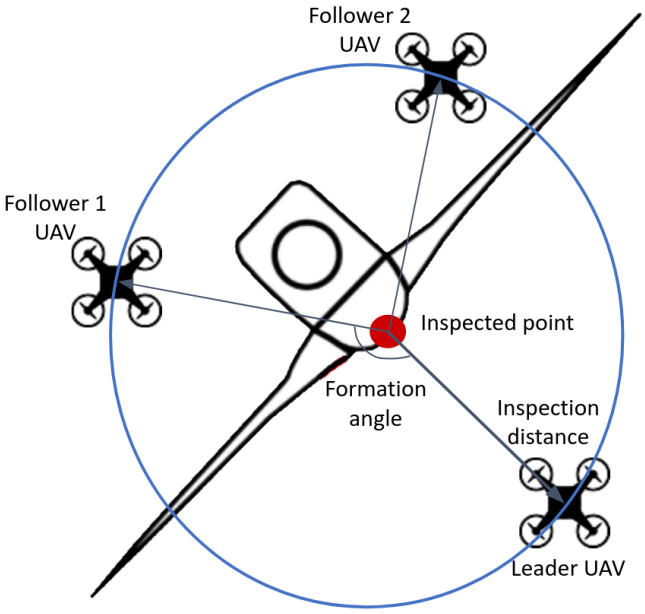

Figura 4.6 Vista 2D de la formación en la tarea de inspección de turbinas de viento. Fuente: [2].

Con la Cabina se busca replicar todas las funcionalidades de la simulación pero en un entorno virtual. Todo este desarrollo se verá en profundidad en la Sección 4.4.

4.4 Entorno de Realidad Virtual en Unity

La Cabina de Realidad Virtual se desarrolla en Unity y su objetivo es parecerse a un *Virtual Cockpit*. Como se comentaba en la Sección 1.1, este Trabajo Fin de Grado surge, en parte, con la idea de realizar el HMI en Realidad Aumentada que se propone como trabajo futuro en [2]. Aun así, se ha

elegido la Realidad Virtual para realizar esta aplicación porque de esta forma se consigue una mayor inmersividad y, además, existe una limitación tecnológica: las gafas Oculus Quest disponibles para este proyecto no cuentan con la tecnología necesaria para utilizar Realidad Aumentada. Esto se resuelve en las nuevas gafas Meta Quest 2 que sí cuentan con esta tecnología [12].

La Cabina Virtual se desarrolla en Unity, pero su implementación se realiza con las gafas Oculus Quest para poder hacer uso de la Realidad Virtual.

Con la adaptación del HMI a un entorno de Realidad Virtual se busca mejorar la forma en que se observan las vistas de las cámaras y la información relativa a la misión. Además, se desarrolla una manera de poder modificar parámetros de la misión de inspección. También se añade una maqueta 3D en el entorno virtual para tener una mejor perspectiva de lo que ocurre en el mundo real.

A continuación, se va a realizar una descripción del entorno en Realidad Virtual de la aplicación. En la Figura 4.7 puede verse el entorno virtual en 3D desarrollado en Unity, donde se ubica la Cabina Virtual y donde tiene lugar el desarrollo de la aplicación. Pueden verse algunos de los elementos que forman parte de la misma, como las pantallas, la mesa de status, el mapa 3D, etc.

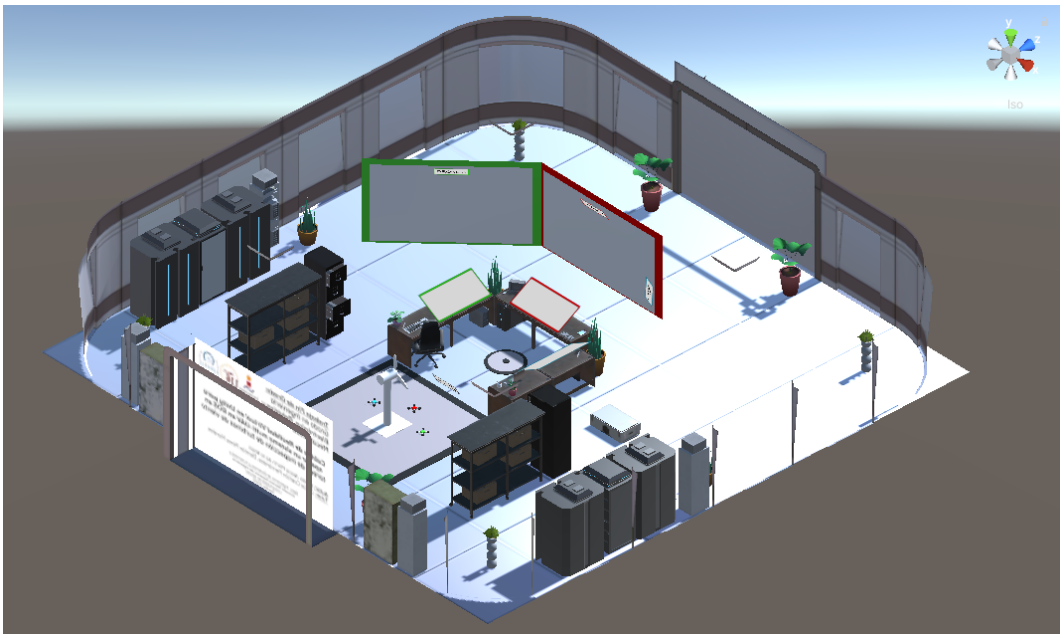

Figura 4.7 Mundo virtual 3D en Unity. Fuente: Elaboración propia.

En la Figura 4.8 se encuentra la Jerarquía completa del entorno de Unity con todos los *GameObject* de la aplicación.

Esta Jerarquía se divide en:

- *SCRIPTS*: Aquí se encuentran todos los scripts de código necesarios para cada una de las funcionalidades de la aplicación. El resto de la Jerarquía solo incluye elementos visuales.
- *3D MAP*: Todo lo relativo a la parte visual del mapa 3D: la cámara para añadir waypoints, el modelo de la turbina de viento y los tres UAVs.
- *SCREENS*: Pantallas de visualización tanto de las vistas de los UAVs como de la información de la misión.
- *ENVIRONMENT*: Elementos visuales relacionados con el entorno virtual. Estos elementos no forman parte de las funcionalidades de la aplicación y su objetivo está más relacionado con la ornamentación y el diseño del ambiente del entorno.

[12] store.facebook.com/products/quest-2

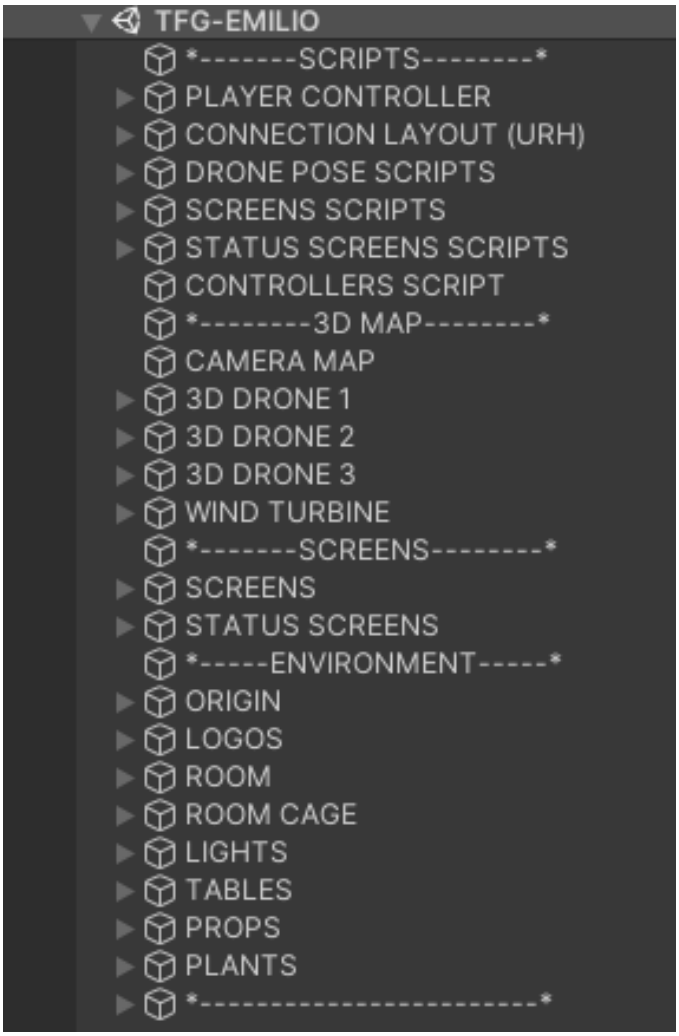

Figura 4.8 Organización del proyecto de Unity. Fuente: Elaboración propia.

Cada uno de estos *GameObject* de la Jerarquía pertenece a una de las cinco partes en las que se estructura la Cabina Virtual (Sección 4.1 y Figura 4.1). Estas cinco partes no se corresponden necesariamente con la estructura de la Jerarquía ya que, por ejemplo, los scripts de la Jerarquía pertenecen cada uno a una de las cinco partes de la estructura.

A continuación, se mencionan las cinco partes de la estructura junto con los *GameObject* que las comprenden:

- ENTORNO VIRTUAL: *CONNECTION LAYOUT (URH)*, *ORIGIN*, *LOGOS*, *ROOM*, *ROOM CAGE*, *LIGHTS*, *TABLES*, *PROPS* y *PLANTS*.
- VISTAS DE LAS CÁMARAS: *SCREENS SCRIPTS* y *SCREENS*.
- MESA DE STATUS Y CONTROL: *STATUS SCREENS SCRIPTS* y *STATUS SCREENS*.
- CONTROLES: *PLAYER CONTROLLER* y *CONTROLLERS SCRIPT*.
- MAPA 3D: *DRONE POSE SCRIPTS*, *CAMERA MAP*, *3D DRONE 1*, *3D DRONE 2*, *3D DRONE 3* y *WIND TURBINE*.

La primera parte, el entorno virtual, engloba todo aquello que no pertenece a alguna de las funcionalidades de la aplicación. Son los *GameObject* que tienen una funcionalidad más estética o auxiliar.

En *CONNECTION LAYOUT (URH)* se encuentra el paquete de Visualizaciones de Unity Robotics Hub (Sección 4.2.1) que permite visualizar en el editor de Unity los mensajes que se intercambian con ROS a través de los distintos topics. *ORIGIN* indica el centro del entorno y la localización de inicio al comenzar a usar la aplicación. *LOGOS* contiene una imagen con información relacionada con el Trabajo Fin de Grado como el autor, los tutores, el título, etc. Esta información puede verse en la pared situada frente a las pantallas de las vistas de las cámaras. *ROOM* contiene el suelo, paredes y techo de la habitación, que es cuadrada, mientras que *ROOM CAGE* es una "jaula" que contiene a la habitación. Su funcionalidad es únicamente delimitar la habitación en el espacio. *LIGHTS*, *TABLES*, *PROPS* y *PLANTS* contienen, respectivamente, las luces, mesas, accesorios y plantas que forman parte del entorno.

En esta sección se ha explicado en profundidad la primera de las cinco partes: el entorno virtual. En las siguientes secciones se desarrollaran en mas detalle las cuatro partes restantes.

Finalmente, en la Figura 4.9 se presenta un plano con la distribución en planta de la escena de Unity. Además, se señalan las cinco partes de la estructura de la Cabina Virtual que se han comentado previamente.

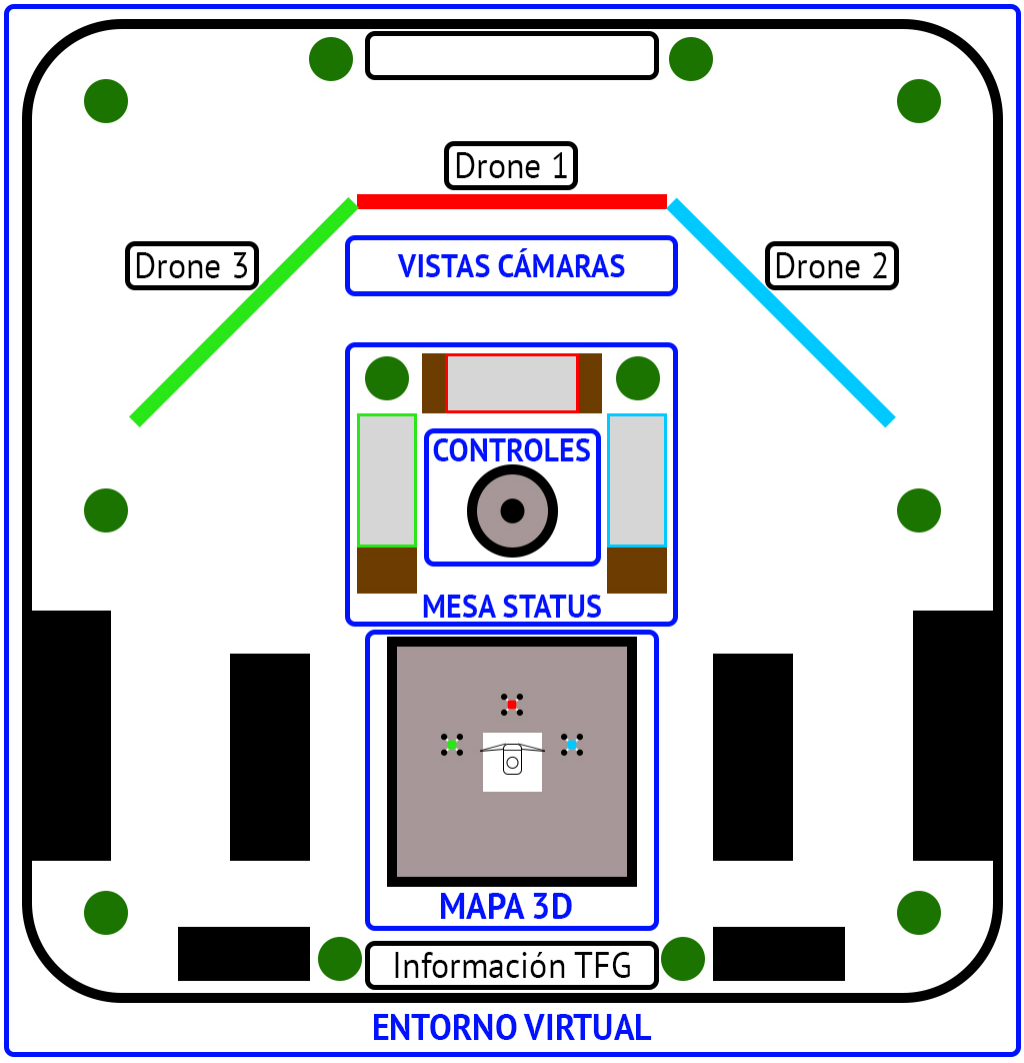

Figura 4.9 Distribución y partes de la Cabina Virtual. Fuente: Elaboración propia.

4.4.1 Vistas de las cámaras de los UAVs

En la Cabina Virtual se sitúan tres pantallas flotantes frente al Mapa 3D para poder ver las vistas de las cámaras de los tres UAVs de la simulación de ROS. Estas pantallas, que son planos de Unity, se ubican en *SCREENS* y cada una de ellas tiene un cartel identificando el UAV al que corresponden las imágenes. Además, cada pantalla tiene un marco con el color de su UAV. Las pantallas de los UAV 2 y 3 se colocan a 45º de la pantalla del UAV 1. Estas pantallas pueden observarse en la Figura 4.10.

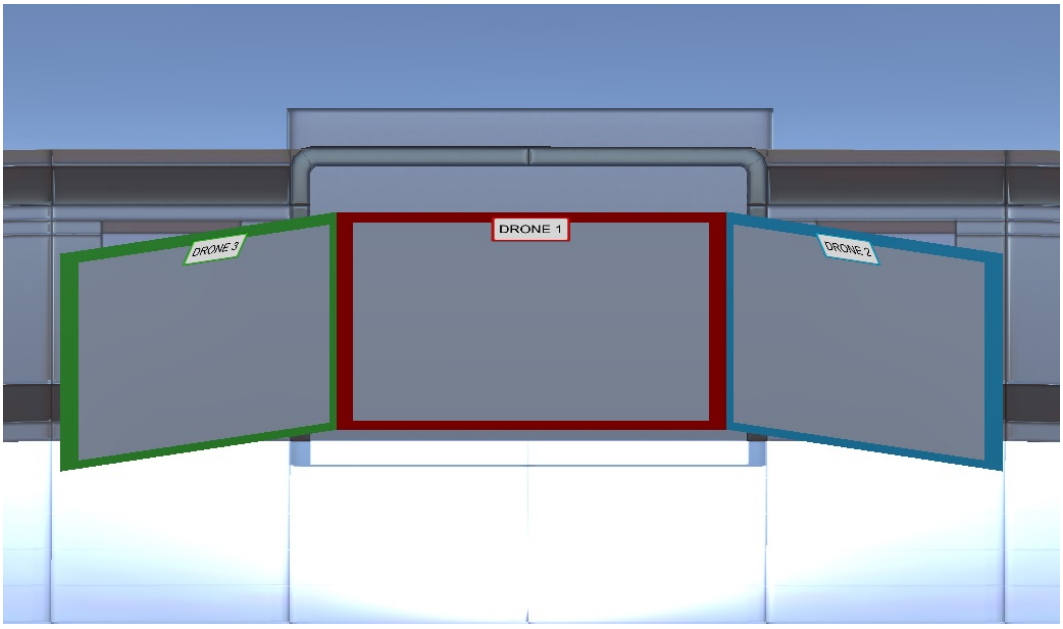

Figura 4.10 Pantallas de las vistas de las cámaras de los UAV. Fuente: Elaboración propia.

En *SCREENS SCRIPTS* se encuentran tres scripts correspondientes a cada una de las pantallas. Gracias a estos scripts las imágenes de los UAVs se envían desde ROS a Unity y son plasmadas en los tres diferentes planos (pantallas) que antes se han explicado.

Los tres scripts son realmente el mismo. El script se llama *ImageSubscriber* y, para que funcione correctamente, se le deben especificar tres parámetros. El primero es el topic en el que se encuentran las imágenes del UAV, en este caso es */mbzirc_X/camera_0/image_raw/compressed*, donde X será 1, 2 o 3 dependiendo del UAV al que haga referencia. El segundo es el Time Step, que por defecto es 0. En el tercero se tiene que indicar el plano de Unity (pantalla) en el que se mostrarán las imágenes.

ImageSubscriber es un script en C# que se suscribe al topic que se le indica y extrae su información, en este caso imágenes, y trata esas imágenes para finalmente aplicarlas como material al plano correspondiente.

En el Pseudocódigo 4.1 se detalla el funcionamiento de *ImageSubscriber*.

```
1. Declarar meshRenderer, texture2D, imageData y isMessageReceived
2. Inicializar texture2D = new Texture2D(1, 1) y meshRenderer.
   material = new Material(Shader.Find("Standard"))
3. Ejecutar ReceiveMessage
   3.1. Inicializar imageData = compressedImage.data; y
      isMessageReceived = true;
4. Bucle de ejecución
   4.1. Espera que isMessageReceived = true para ejecutar
      ProcessMessage
```

```
            4.1.1. Reemplazar imageData en texture2D
            4.1.2. Aplicar cambios de texture2D
            4.1.3. Establecer texture2D en meshRenderer.material
            4.1.4. isMessageReceived = false
        4.2. Espera recibir imagen del tópico en la función callback
            4.2.1. isMessageReceived = true
```

Código 4.1 Pseudocódigo de *ImageSubscriber*.

4.4.2 Mesa de status y control

En la Figura 4.11, se pueden apreciar las tres mesas correspondientes a los tres UAVs de la simulación de ROS. Se colocan dispuestas en la parte central del entorno. En cada mesa, se encuentra una pantalla de características muy similares a las de la Sección 4.4.1. Estas pantallas se encuentran en *STATUS SCREENS* y tienen como función mostrar información y datos relevantes de cada uno de los UAVs y de la misión.

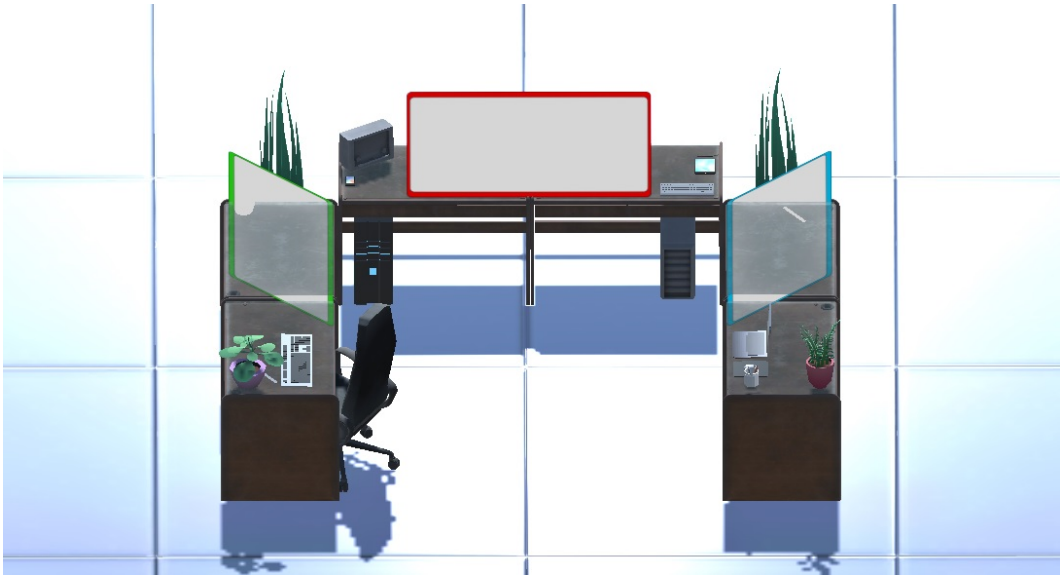

Figura 4.11 Mesas de status y control de los tres UAVs. Fuente: Elaboración propia.

En cada pantalla aparece el siguiente texto *"Welcome to the DURABLE's Virtual Cockpit Interface!"*. Justo debajo aparece la hora y después se indica el UAV del que se está mostrando la información. Además, se indica si el UAV es líder o seguidor y su estado. Después, se muestra la posición y la distancia de inspección. En caso de que el UAV sea seguidor se añade el ángulo de formación. A continuación, se indican cuáles son las referencias de distancia de inspección y de ángulo de formación en ese momento. Finalmente, se indica el modo de operación del UAV, que puede ser: Non-stopping, Smooth, Inspecting y Stopping.

En *STATUS SCREENS SCRIPTS* se encuentran los scripts que hacen todo esto posible. El script es genérico para cada UAV, pudiendo discernir de un UAV líder y uno seguidor para plasmar diferente información. El script lee los tópicos relacionados con los datos e información de la misión y los presenta en tres pantallas distintas, una para cada UAV.

El script se llama *Status_Screen* y, para que funcione correctamente, hay que indicarle desde Unity para qué UAV se va a ejecutar y si este es líder o seguidor. También hay que asignar el *GameObject* de tipo texto donde finalmente se mostrará toda la información. Por último, hay que

indicar todos los topics de los que el script extraerá información. Esto puede verse en la Tabla 4.1 donde X será 1, 2 o 3 dependiendo del UAV al que haga referencia.

Tabla 4.1 Topics de información usados por *Status_Screen* .

Tipo	Nombre
Drone_pose_topic	/drone_X/ual/pose
Drone_state_topic	/drone_X/ual/state
Current_inspection_distance	/drone_X/mission_planner_ros/inspection_distance
Reference_inspection_distance	/drone_1/mission_planner_ros/current_distance_respect_the_inspection_point
Current_formation_angle	/drone_X/mission_planner_ros/formation_angle
Reeerence_formation_angle	/drone_1/mission_planner_ros/current_angle_respect_the_inspection_point
Operation_mode_topic	/drone_1/mission_planner_ros/operation_mode

En el Pseudocódigo 4.2 se detalla el funcionamiento de *Status_Screen*.

```
1. Declarar variables, topics y datos
2. Crear conexión WebSocket
3. Inicializar variables auxiliares de texto
4. Inicializar variables de texto con datos de los topics
5. Asignar a text_element la variable de texto final
```

Código 4.2 Pseudocódigo de *Status_Screen*.

4.4.3 Controles de la aplicación

Hasta ahora, la comunicación de Unity con ROS ha sido unidireccional. Se han recibido imágenes de las cámaras e información y datos de la misión, pero no se ha enviado nada a ROS. Para poder operar la misión e interactuar con la Cabina Virtual se hace uso de los controles Oculus Touch (Sección 3.3).

En *PLAYER CONTROLLER* se encuentra *OVRPlayerController* del paquete *Oculus Integration*. *OVRPlayerController* contiene una cámara y un personaje adaptados a la Realidad Virtual, lo que permite moverse dentro del entorno virtual. Además, configura los controles Oculus Touch, por lo que facilita la integración de las gafas y los mandos con el entorno virtual.

Con los controles se pretende poder operar la formación multi-UAV haciendo uso de comandos de alto nivel. Se busca poder modificar parámetros como la distancia de inspección, el ángulo de formación o el tiempo de órbita. Además, se quiere poder despegar y aterrizar los UAVs, parar e iniciar la misión, entre otras cosas. Esto se correspondería con el Modo 1. El Modo 2, se utiliza para añadir o eliminar waypoints a la trayectoria de la misión y, para ello, se recurre al Mapa 3D.

A continuación, se presenta la Tabla 4.2, donde se define la funcionalidad de los botones, gatillos y joysticks en la Cabina Virtual según el modo de operación. En la Figura 3.5, se pueden apreciar los distintos elementos que componen los mandos.

Para realizar todo esto, se hace uso de *ControllersDURABLE*, que se encuentra en *CONTROLLERS SCRIPT*. *ControllersDURABLE* es un script en el que se configura la cámara del personaje, la del mapa 3D y la ubicación de la turbina. Además se configuran algunos parámetros adicionales y los topics de distancia, ángulo y tiempo de órbita. Este script cuenta con dos modos. En el Modo 1

Tabla 4.2 Funciones de los botones, gatillos y joysticks de los controles.

	Modo 1	Modo 2
Botón X	Aterrizar	Aterrizar
Botón Y	Despegar	Despegar
Botón A	Parar misión	Parar misión
Botón B	Iniciar misión	Iniciar misión
Gatillo Lateral I	Cambio Modo 2	Cambio Modo 1
Gatillo Trasero I	-	Añadir waypoints
Gatillo Lateral D	-	-
Gatillo Trasero D	Home	Borrar waypoints
Joystick I	Cambia tiempo órbita (Vertical)	Mover cámara mapa
Joystick D	Cambia distancia (Vertical) y ángulo (Horizontal)	-

(*mode_selector* = *false*) se puede operar la misión y realizar distintas tareas como puede verse en la Tabla 4.2. En el Modo 2 (*mode_selector* = *true*), se cambia la vista del personaje por la de la cámara y mediante los mandos se pueden añadir o quitar waypoints a la trayectoria de los UAVs.

En el Pseudocódigo 4.3 se profundiza en el funcionamiento de *ControllersDURABLE*.

```
1. Declarar variables
2. Crear conexión WebSocket
3. Si A: Parar misión, B: Iniciar misión, X: Aterrizar, Y: Despegar
4. Si LeftSideTrigger: mode_selector = !mode_selector
5. Si mode_selector = true: Modo 2
   5.1. Si LeftBackTrigger: Añadir waypoint
   5.2. Si RightBackTrigger: Borrar waypoints
   5.3. Si LeftJoystick: Mover cámara mapa
6. Si mode_selector = false: Modo 1
   6.1. Si RightBackTrigger: Home
   6.2. Si RightJoystick: Cambiar distancia (Vertical) y ángulo (
      Horizontal)
   6.3. Si LeftJoystick: Cambiar tiempo órbita (Vertical)
```

Código 4.3 Pseudocódigo de *ControllersDURABLE*.

4.4.4 Mapa 3D

En la Cabina Virtual no se tiene acceso, en teoría, a otras fuentes de información que no provengan de la propia Realidad Virtual. Además, en una implementación real la perspectiva del operario no siempre es favorable. Es por ello que se añade un Mapa 3D en la Cabina Virtual para representar lo que ocurre en el escenario real y aumentar la inmersividad en la aplicación. Además, esta maqueta se usa en el Modo 2 de los controles de la aplicación para seleccionar la posición de los waypoints a incluir. En este modo se usa la cámara del mapa y se puede modificar la altura y el ángulo respecto a la turbina.

En la Figura 4.12, se observa la maqueta 3D formada por una turbina de viento y tres UAVs, cada uno de ellos con su color asignado. Los modelos 3D de la cámara del mapa, los UAVs y la turbina de viento se encuentran en *CAMERA MAP*, *3D DRONE 1*, *3D DRONE 2*, *3D DRONE 3* y *WIND TURBINE*, respectivamente.

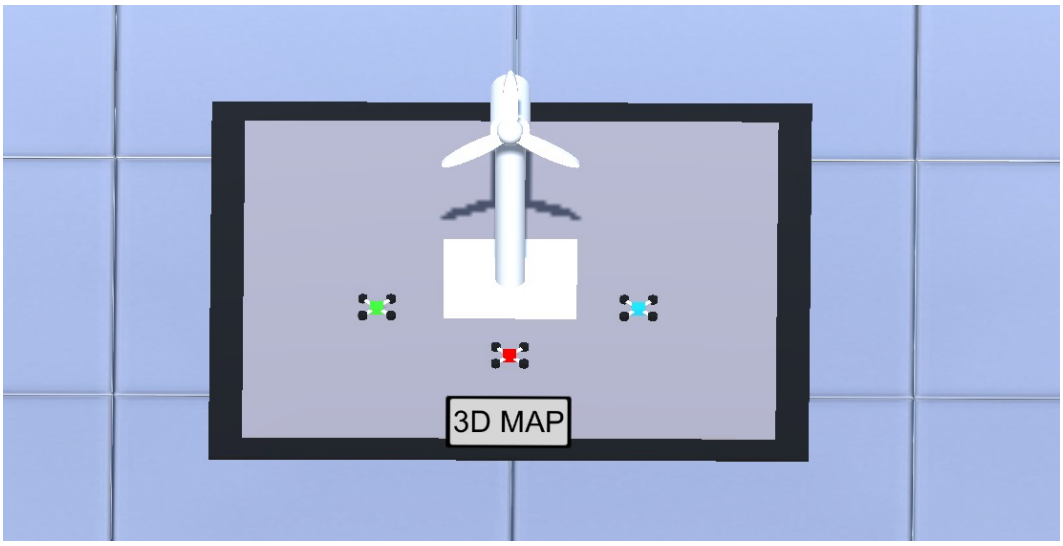

Figura 4.12 Mapa 3D. Fuente: Elaboración propia.

En *DRONE POSE SCRIPTS* se encuentra el script *Drone_Pose*. Este script permite la recreación en Unity de la simulación que se ejecuta en ROS. El mapa 3D obtiene las posiciones de los UAVs y replica su comportamiento. Estos datos son tratados y escalados, ya que se trata de una representación de la realidad.

Para ejecutar *Drone_Pose* se especifica la posición de la turbina y del UAV líder en Unity. Además, hay que añadir el topic en el que se encuentra la información de la posición. También, se pueden modificar los parámetros de escala.

En el Pseudocódigo 4.4 se explica el funcionamiento de *Drone_Pose*.

```
1. Declarar variables
2. Crear conexión WebSocket
    2.1. Obtener pose de ROS
    2.2. isMessageReceived = true
3. Si isMessageReceived = true
    3.1. Ejecutar Scale: Posición de ROS escalada
    3.2. Ejecutar TranslateToWindTurbine: Posición trasladada
    3.3. Ejecutar ProcessMessage: Asignar pose trasladada a pose en
       Unity
```

Código 4.4 Pseudocódigo de *Drone_Pose*.

5 Resultados

El proyecto completo de este Trabajo Fin de Grado con todos los scripts de código, paquetes, librerías, y un vídeo ilustrativo de los resultados del mismo, puede encontrarse online [1].

Los resultados de este proyecto se presentan por medio de un vídeo en el cual se realiza una demostración en vivo del funcionamiento de la Cabina Virtual. A continuación, se procede a explicar los resultados.

En primer lugar, en la Figura 5.1 y Figura 5.2, se muestra el entorno de la cabina. Se pueden observar las pantallas de las cámaras y las mesas de status. Además, en la parte trasera se encuentra el mapa 3D.

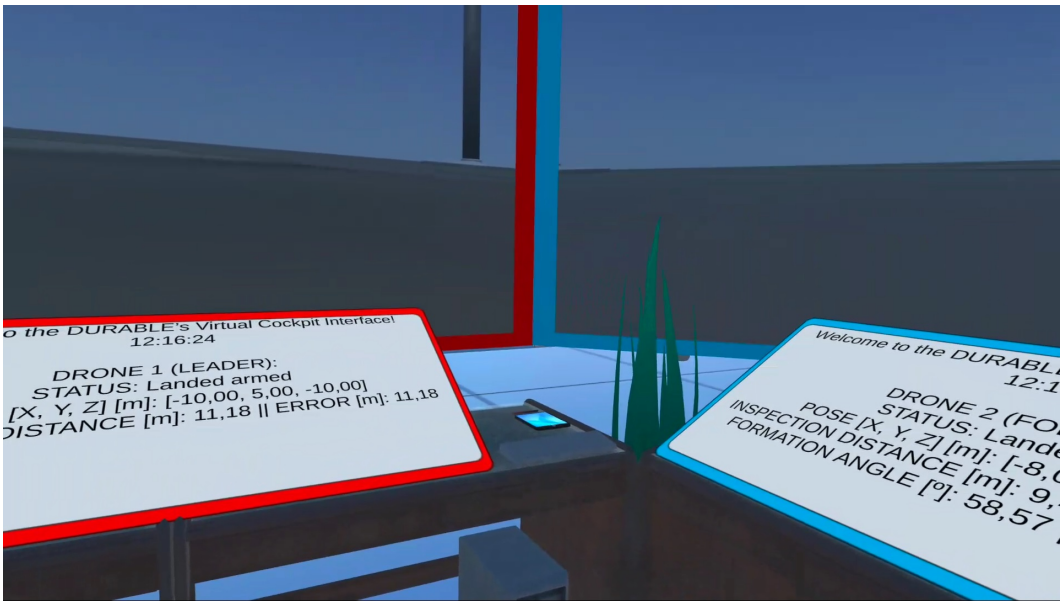

Figura 5.1 Presentación del entorno. Fuente: Elaboración propia.

A continuación, se procede a añadir waypoints a la trayectoria de los UAVs. Se activa el Modo 2 donde se cambia a la cámara del mapa, y mediante los controles, el usuario se puede desplazar alrededor de la turbina y añadir diferentes waypoints. (Figura 5.3)

Tras esto, se vuelve al Modo 1 (Figura 5.4) y se despegan los UAVs y se inicia la misión.

En la Figura 5.5 y Figura 5.6, se puede confirmar el correcto funcionamiento del sistema. Los UAVs siguen la trayectoria definida por los waypoints anteriores.

[1] Repositorio del proyecto

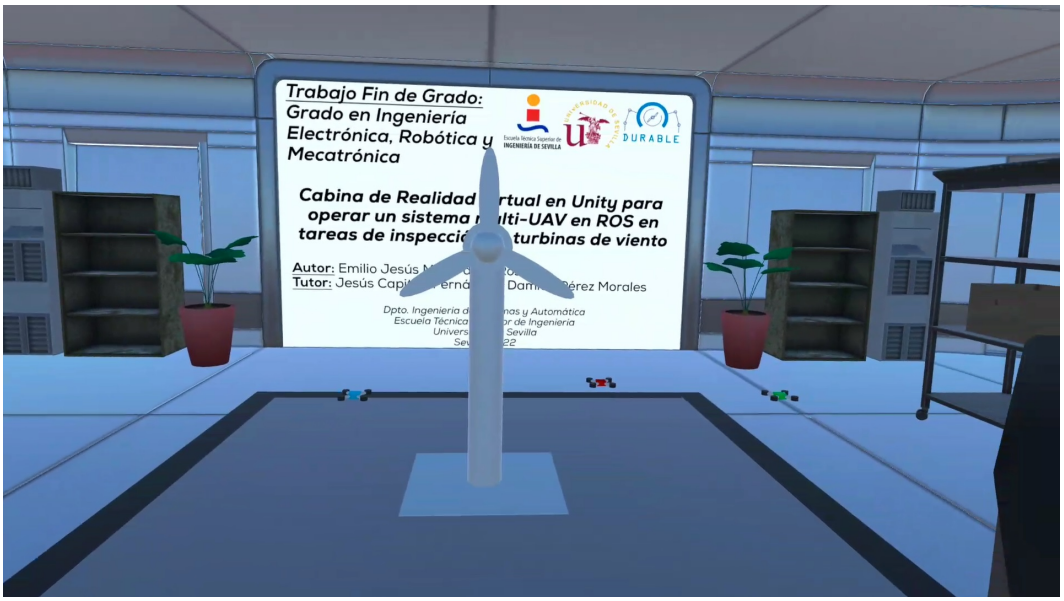

Figura 5.2 Presentación del entorno. Fuente: Elaboración propia.

Figura 5.3 Adición de waypoints. Fuente: Elaboración propia.

Comparando las dos imágenes se puede confirmar que, efectivamente, lo que se ve en la pantalla de cada UAV se corresponde con la posición real de los UAVs.

Por último, en la Figura 5.7, se cambia la distancia de inspección y la formación se adapta a esta nueva distancia.

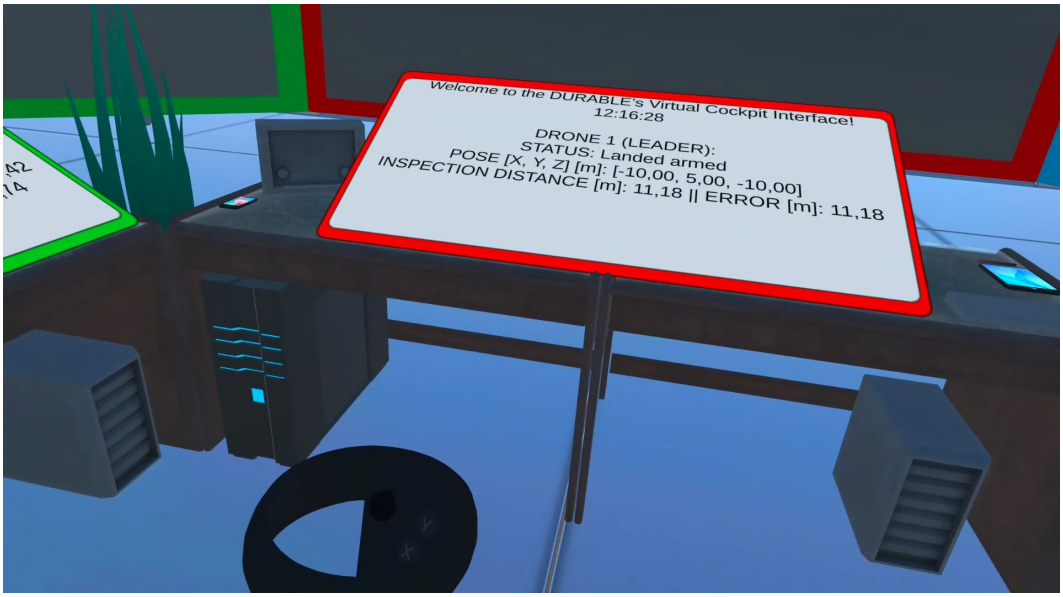

Figura 5.4 Inicio de la misión. Fuente: Elaboración propia.

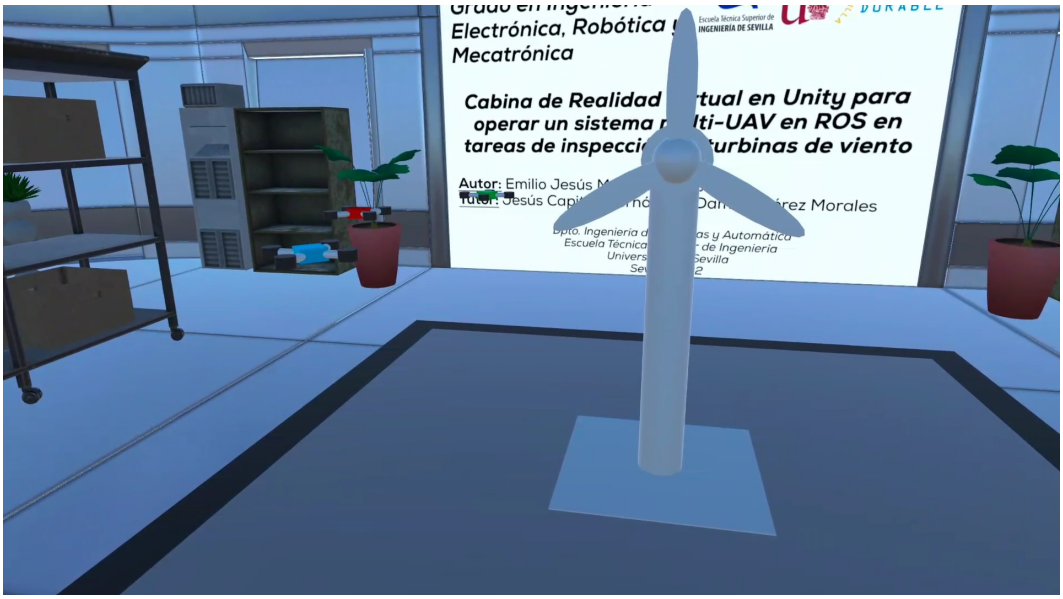

Figura 5.5 Funcionamiento de la misión. Fuente: Elaboración propia.

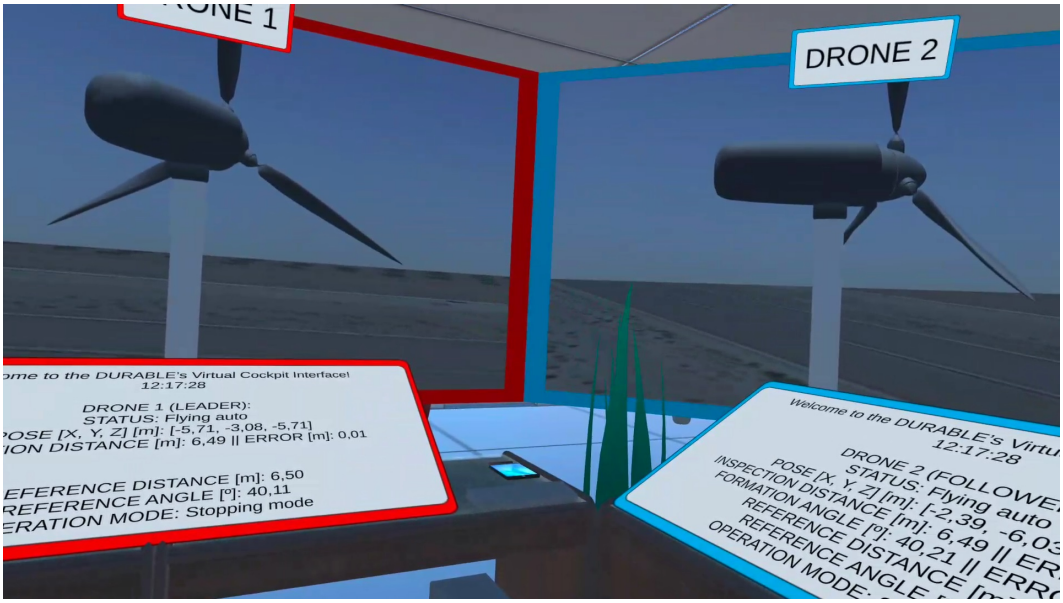

Figura 5.6 Funcionamiento de la misión. Fuente: Elaboración propia.

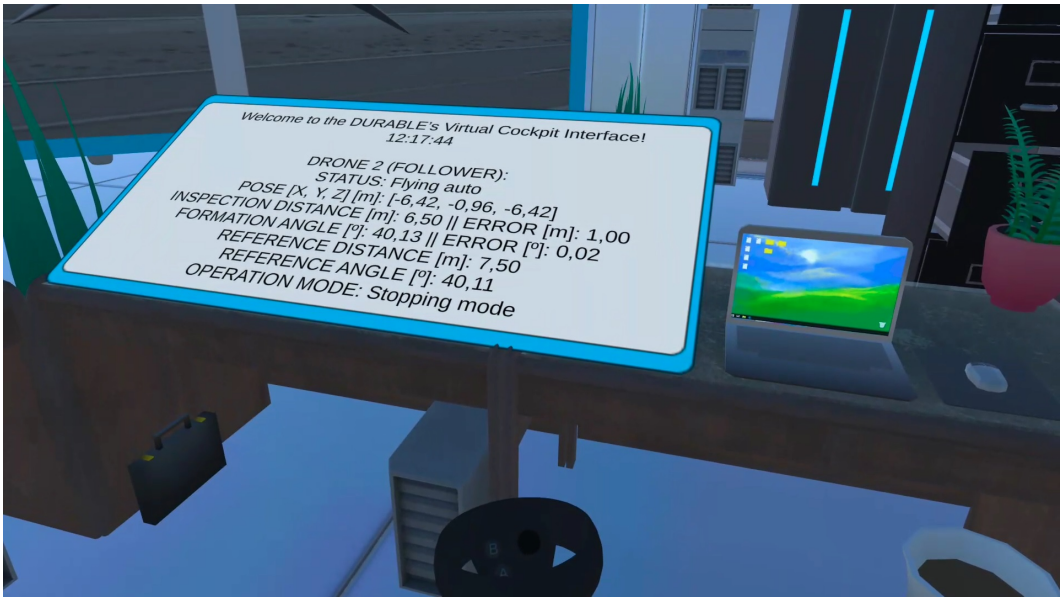

Figura 5.7 Cambio de la distancia de referencia. Fuente: Elaboración propia.

6 Conclusiones y líneas futuras

En este proyecto se ha mostrado el desarrollo de una Cabina de Realidad Virtual para operar sistemas multi-UAV. Para cumplir este objetivo la Cabina se ha desarrollado en Unity mientras que la simulación de la tarea de inspección se ha ejecutado en ROS. Esto ha requerido desarrollar comunicaciones entre Unity y ROS haciendo uso de ROS#. En la Cabina Virtual se ha conseguido visualizar las vistas de las cámaras de los UAVs. Además, se ha podido presentar en pantallas de status información relevante de la misión de inspección. En cuanto a los controles, se ha conseguido implementar en los mandos Oculus Touch las funcionalidades requeridas por la Cabina Virtual. También se ha creado un mapa 3D que emula dentro de la Cabina Virtual el entorno real donde se desarrolla la tarea de inspección.

Se puede afirmar entonces que la Cabina ha cumplido los objetivos inicialmente propuestos de manera satisfactoria. Se ha conseguido desarrollar una aplicación en Realidad Virtual que permite a un usuario operar un sistema de UAVs desde el "mundo virtual". Al contrario que las interfaces tradicionales, esta novedosa aplicación consigue una mayor interacción e inmersividad por parte del usuario. Cabe mencionar que esta Cabina Virtual tiene una aplicabilidad real. Se ha desarrollado en un entorno de simulación, pero sin duda, la adaptabilidad a una aplicación real sería muy sencilla.

El mundo de la Realidad Virtual es enorme y ofrece muchísimas posibilidades. Es complicado decidir como se podría seguir mejorando esta aplicación, pues las opciones son inmensas. Aún así se plantean algunas ideas que se podrían implementar en el futuro:

- Transformar el proyecto a Realidad Mixta, es decir, añadir Realidad Aumentada.
- Añadir un HUD que muestre información relevante, de forma que siempre esté presente en la aplicación.
- Añadir algún sistema que mejore la adición de waypoints, como la visualización en el mapa 3D de estos waypoints.

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] J. Roldán Gómez, E. Peña-Tapia, M. Garzon, A. Martín-Barrio, and A. Barrientos, "Interfaz de control para un robot manipulador mediante realidad virtual," 09 2017.

[2] D. Pérez, A. Alcántara, and J. Capitán, "Distributed trajectory planning for a formation of aerial vehicles inspecting wind turbines," 2022.

[3] J. F. Keane and S. S. Carr, "A brief history of early unmanned aircraft," *Johns Hopkins APL Technical Digest*, vol. 32, no. 3, pp. 558–571, 2013.

[4] P. G. Fahlstrom, T. J. Gleason, and M. H. Sadraey, *Introduction to UAV systems*. John Wiley & Sons, 2022.

[5] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.

[6] E. Semsch, M. Jakob, D. Pavlicek, and M. Pechoucek, "Autonomous uav surveillance in complex urban environments," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 2, 2009, pp. 82–85.

[7] K. T. San, S. J. Mun, Y. H. Choe, and Y. S. Chang, "Uav delivery monitoring system," in *MATEC Web of Conferences*, vol. 151. EDP Sciences, 2018, p. 04011.

[8] R. La Scalea, M. Rodrigues, D. P. M. Osorio, C. H. Lima, R. D. Souza, H. Alves, and K. C. Branco, "Opportunities for autonomous uav in harsh environments," in *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, 2019, pp. 227–232.

[9] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, "Lsar: Multi-uav collaboration for search and rescue missions," *IEEE Access*, vol. 7, pp. 55 817–55 832, 2019.

[10] V. Krátký, A. Alcántara, J. Capitán, P. Štěpán, M. Saska, and A. Ollero, "Autonomous aerial filming with distributed lighting by a team of unmanned aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7580–7587, 2021.

[11] D. Casbeer, R. Beard, T. McLain, S.-M. Li, and R. Mehra, "Forest fire monitoring with multiple small uavs," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, pp. 3530–3535 vol. 5.

[12] C. Yuan, Z. Liu, and Y. Zhang, "Uav-based forest fire detection and tracking using image processing techniques," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015, pp. 639–643.

[13] M. Gschwindt, E. Camci, R. Bonatti, W. Wang, E. Kayacan, and S. Scherer, "Can a robot become a movie director? learning artistic principles for aerial cinematography," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1107–1114.

[14] V. P. Subba Rao and G. S. Rao, "Design and modelling of anaffordable uav based pesticide sprayer in agriculture applications," in *2019 Fifth International Conference on Electrical Energy Systems (ICEES)*, 2019, pp. 1–4.

[15] D. Popescu, F. Stoican, G. Stamatescu, L. Ichim, and C. Dragana, "Advanced uav–wsn system for intelligent monitoring in precision agriculture," *Sensors*, vol. 20, no. 3, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/3/817

[16] A. Kulsinskas, P. Durdevic, and D. Ortiz-Arroyo, "Internal wind turbine blade inspections using uavs: Analysis and design issues," *Energies*, vol. 14, no. 2, 2021. [Online]. Available: https://www.mdpi.com/1996-1073/14/2/294

[17] M. Alsafasfeh, I. Abdel-Qader, B. Bazuin, Q. Alsafasfeh, and W. Su, "Unsupervised fault detection and analysis for large photovoltaic systems using drones and machine vision," *Energies*, vol. 11, no. 9, p. 2252, 2018.

[18] G. Silano, J. Bednar, T. Nascimento, J. Capitan, M. Saska, and A. Ollero, "A multi-layer software architecture for aerial cognitive multi-robot systems in power line inspection tasks," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2021, pp. 1624–1629.

[19] J. Huuskonen and T. Oksanen, "Soil sampling with drones and augmented reality in precision agriculture," *Computers and Electronics in Agriculture*, vol. 154, pp. 25–35, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169918301650

[20] S. Ruano, C. Cuevas, G. Gallego, and N. García, "Augmented reality tool for the situational awareness improvement of uav operators," *Sensors*, vol. 17, no. 2, p. 297, 2017.

[21] J. Ernst, N. Peinecke, L. Ebrecht, S. Schmerwitz, and H.-U. Döhler, "Virtual cockpit: An immersive head-worn display as humanmachine interface for helicopter operations," *Optical Engineering*, vol. 58, 05 2019.

[22] I. Kalinov, D. Trinitatova, and D. Tsetserukou, "Warevr: Virtual reality interface for supervision of autonomous robotic system aimed at warehouse stocktaking," 10 2021.

[23] S. S. Esfahlani, "Mixed reality and remote sensing application of unmanned aerial vehicle in fire and smoke detection," *Journal of Industrial Information Integration*, vol. 15, pp. 42–49, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2452414X18300773

[24] M. P. Das, Z. Dong, and S. Scherer, "Joint point cloud and image based localization for efficient inspection in mixed reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6357–6363.

[25] J. Linowes, *Unity virtual reality projects*. Packt Publishing Ltd, 2015.